ROGUE WAVE®
SOFTWARE

2012 Programming weather, climate, and earth-system models on heterogeneous multi-core platforms

September 12-13, 2012 at the National Center for Atmospheric Research in Boulder, Colorado

Addressing the Increasing Challenges of Debugging on Accelerated Multi-Core Systems

Ed Hinkel
Senior Sales Engineer

Agenda

- Overview - Rogue Wave & TotalView
- Heterogeneous Systems - Then and Now
- Debugging Accelerated Systems
  - GPU - Nvdia CUDA
  - MIC - Intel Phi

Rogue Wave Today

The largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications

Leader in embeddable math and statistics algorithms and visualization software for data-intensive applications.

Leading provider of intelligent software technology which analyzes and optimizes computing performance in single and multi-core environments.

Industry-leading interactive analysis and debugging tools for the world's most sophisticated software applications.

Latest addition to the Rogue Wave family: Rogue Wave Visualization for C++
(Formerly IBM's ILOG Visualization for C++ products)

Representative Customers

BARCLAYS CAPITAL Morgan Stanley Deutsche Bank UBS

Bank of America JPMorganChase citi MOTOROLA

PIXAR NORTHROP GRUMMAN bmcsoftware ارامكو السعودية Saudi Aramco

Disney at&t Qwest NOKIA BT

Blue Sky verizon ERICSSON HUAWEI

RiskMetrics Group
The Center for the Financial Community

Deutsche Telekom NASA BOEING 3DS SIMULIA

Raytheon Sandia National Laboratories

LOCKHEED MARTIN Lawrence Livermore National Laboratory

MIZUHO ExxonMobil SIEMENS

ANVITA HEALTH™
Actionable Health Intelligence™

CISCO IBM OPENWAVE™

vodafone SYBASE® ORACLE® AVAYA SAP

Los Alamos NATIONAL LABORATORY BIZERBA ... closer to your business ChevronTexaco

ROGUE WAVE® SOFTWARE

Heterogeneous Systems - Then and Now

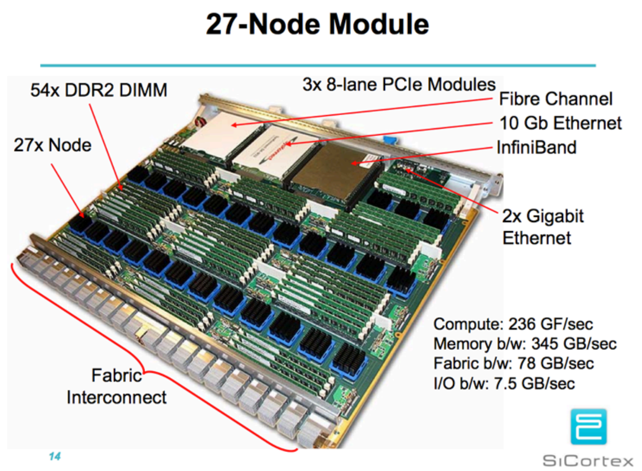
Heterogeneous Systems and the Need for Speed

A couple of pioneers

Heterogeneous Systems - Then and Now

Remember SiCortex?

Heterogeneous Systems - SiCortex 2003 - May 2009

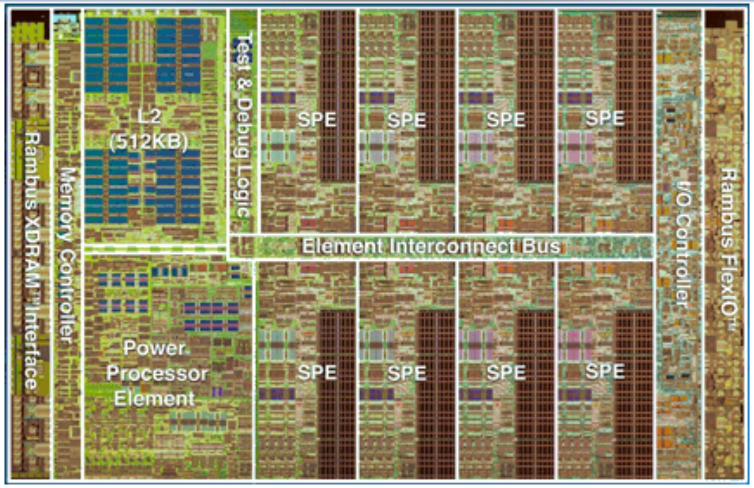


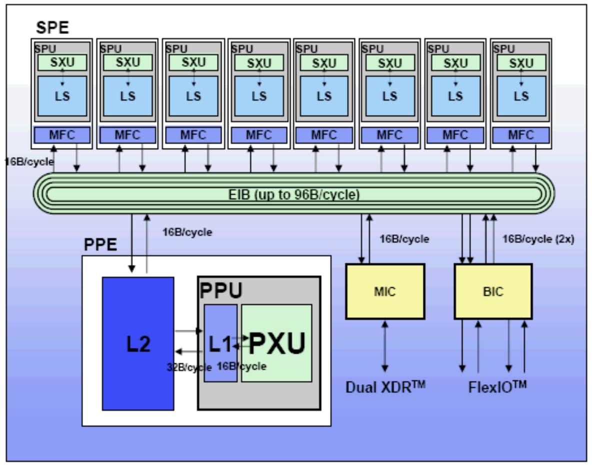
An X-86 host with up to 972 MIPS nodes, with up to 5,832 cores and 7,776 GB of memory

Heterogeneous Systems - Then and Now

Remember Cell?

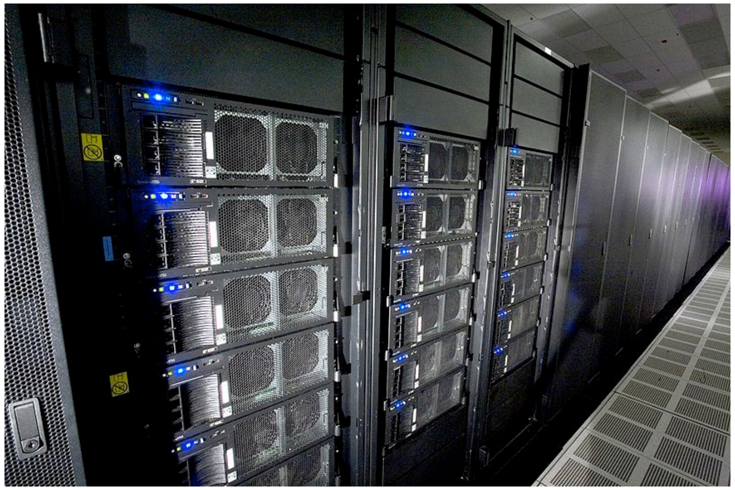
Heterogeneous Systems - CELL March 2001 - Nov 2009





Source: M. Gschwind et al., Hot Chips-17, August 2005

RoadRunner

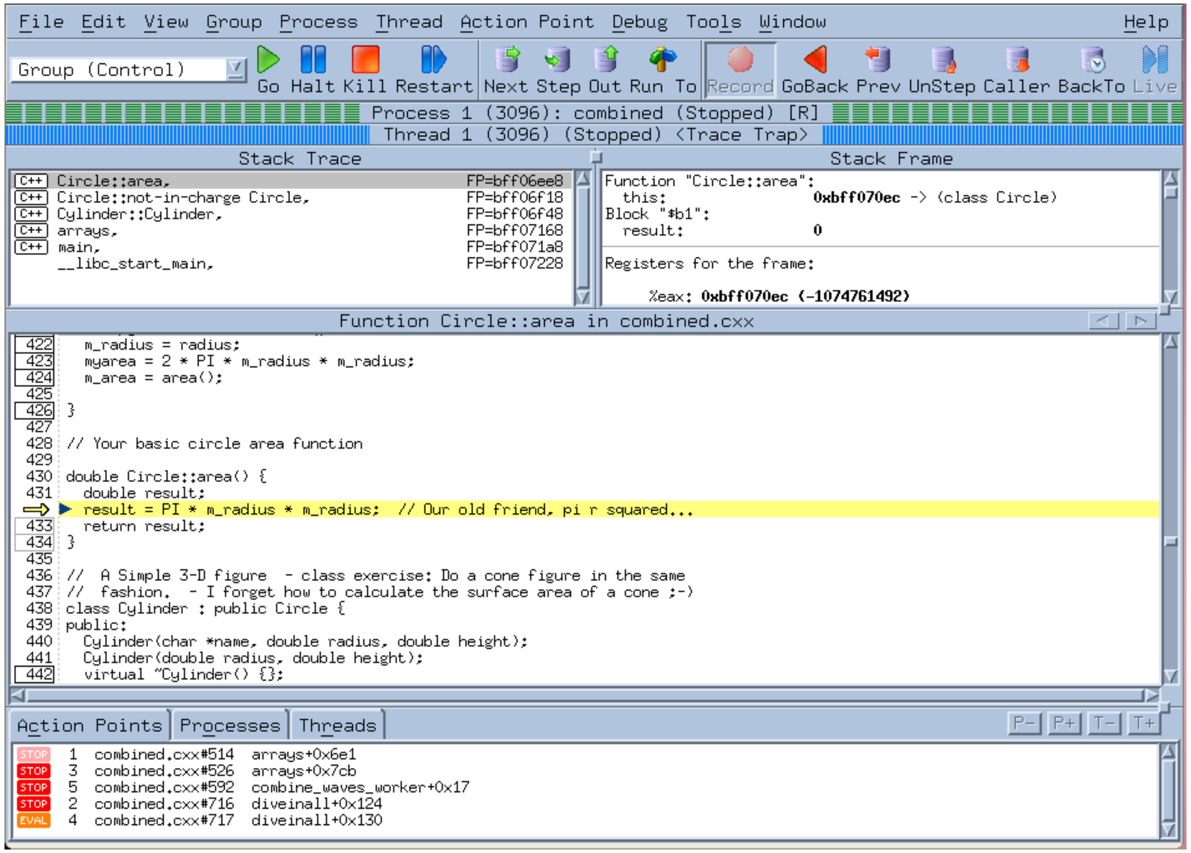


a hybrid design with 12,960 IBM PowerXCell 8i and 6,480 AMD Opteron dual-core processors

What is TotalView?

A comprehensive debugging solution for demanding parallel, heterogeneous, and multi-core applications

- **Wide compiler & platform support**
  - C, C++, Fortran 77 & 90, UPC
  - Unix, Linux, OS X
- **Handles Concurrency**
  - Multi-threaded Debugging
  - Multi-process Debugging
- **Integrated Memory Debugging**
- **Reverse Debugging for Linux**
- **Supports Multiple Usage Models**
  - Powerful and Easy GUI – Highly Graphical
  - CLI for Scripting
  - Long Distance Remote Debugging
  - Unattended Batch Debugging



```
Process 1 (3096): combined (Stopped) [R]
Thread 1 (3096) (Stopped) <Trace Trap>
```

```
C++  Circle::area,                      FP=bff06ee8
C++  Circle::not-in-charge Circle,      FP=bff06f18
C++  Cylinder::Cylinder,                FP=bff06f48
C++  arrays,                            FP=bff07168
C++  main,                              FP=bff071a8
     __libc_start_main,                 FP=bff07228
```

```
Function "Circle::area":
  this:                0xbff070ec -> (class Circle)
Block "$b1":
  result:              0

Registers for the frame:

      %eax: 0xbff070ec (-1074761492)
```

Function Circle::area in combined.cxx

```
  m_radius = radius;
  myarea = 2 * PI * m_radius * m_radius;
  m_area = area();

}

// Your basic circle area function

double Circle::area() {
  double result;
  result = PI * m_radius * m_radius;  // Our old friend, pi r squared...
  return result;
}

//  A Simple 3-D figure  - class exercise: Do a cone figure in the same
//  fashion.  - I forget how to calculate the surface area of a cone ;-)
class Cylinder : public Circle {
public:
  Cylinder(char *name, double radius, double height);
  Cylinder(double radius, double height);
  virtual ~Cylinder() {};
```

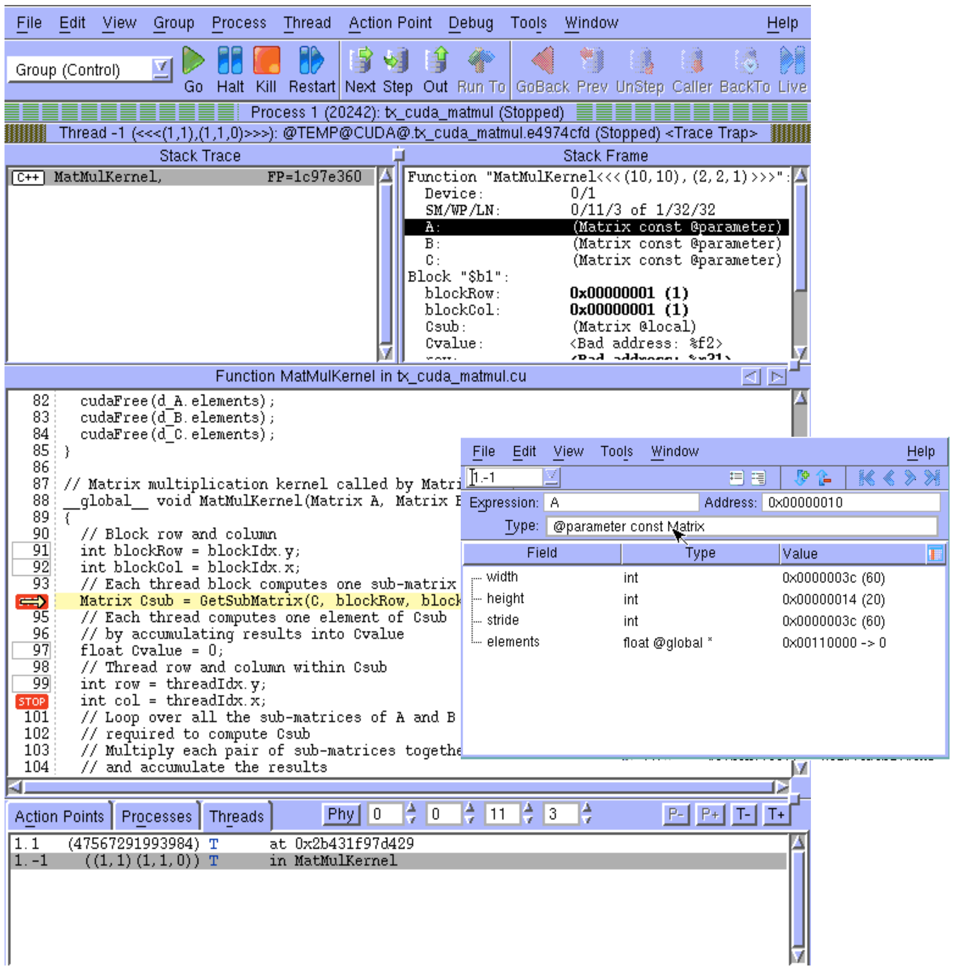
```
STOP  1  combined.cxx#514  arrays+0x6e1
STOP  3  combined.cxx#526  arrays+0x7cb
STOP  5  combined.cxx#592  combine_waves_worker+0x17
STOP  2  combined.cxx#716  diveinall+0x124
EVAL  4  combined.cxx#717  diveinall+0x130
```

GPU Debugging

with

TotalView

CUDA Port of TotalView



File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live
Stack Trace
Stack Frame
Action Points Processes Threads
Phy 0 0 11 3
P- P+ T- T+
File Edit View Tools Window Help

Process 1 (20242): tx_cuda_matmul (Stopped)

Thread -1 (<<<(1,1),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.e4974cfd (Stopped) <Trace Trap>

```
C++  MatMulKernel,                FP=1c97e360
```

```
Function "MatMulKernel<<<(10,10),(2,2,1)>>>":
  Device:              0/1
  SM/WP/LN:            0/11/3 of 1/32/32
  A:                   (Matrix const @parameter)
  B:                   (Matrix const @parameter)
  C:                   (Matrix const @parameter)
Block "$b1":
  blockRow:            0x00000001 (1)
  blockCol:            0x00000001 (1)
  Csub:                (Matrix @local)
  Cvalue:              <Bad address: %f2>
```

Function MatMulKernel in tx_cuda_matmul.cu

```
  cudaFree(d_A.elements);
  cudaFree(d_B.elements);
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by Matri
__global__ void MatMulKernel(Matrix A, Matrix B
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix
  Matrix Csub = GetSubMatrix(C, blockRow, block
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
  // Loop over all the sub-matrices of A and B
  // required to compute Csub
  // Multiply each pair of sub-matrices togethe
  // and accumulate the results
```

```
1.1    (47567291993984) T     at 0x2b431f97d429
1.-1     ((1,1)(1,1,0)) T     in MatMulKernel
```

1.-1

Expression: A   Address: 0x00000010

Type: @parameter const Matrix

Field	Type	Value
width	int	0x0000003c (60)
height	int	0x00000014 (20)
stride	int	0x0000003c (60)
elements	float @global *	0x00110000 -> 0

Full visibility of both Linux and GPU threads

- Device threads shown as part of the parent Unix process
- Handles all the differences between the CPU and GPU

Fully represent the hierarchical memory

- Display data at any level (registers, local, block, global or host memory)
- Making it clear where data resides with type qualification

Thread and Block Coordinates

- Built in runtime variables display threads in a warp, block and thread dimensions and indexes
- Displayed on the interface in the status bar, thread tab and stack frame

Device thread control

- Warps advance synchronously

Handles CUDA function inlining

- Step into or over inlined functions
- Functions show on stack trace

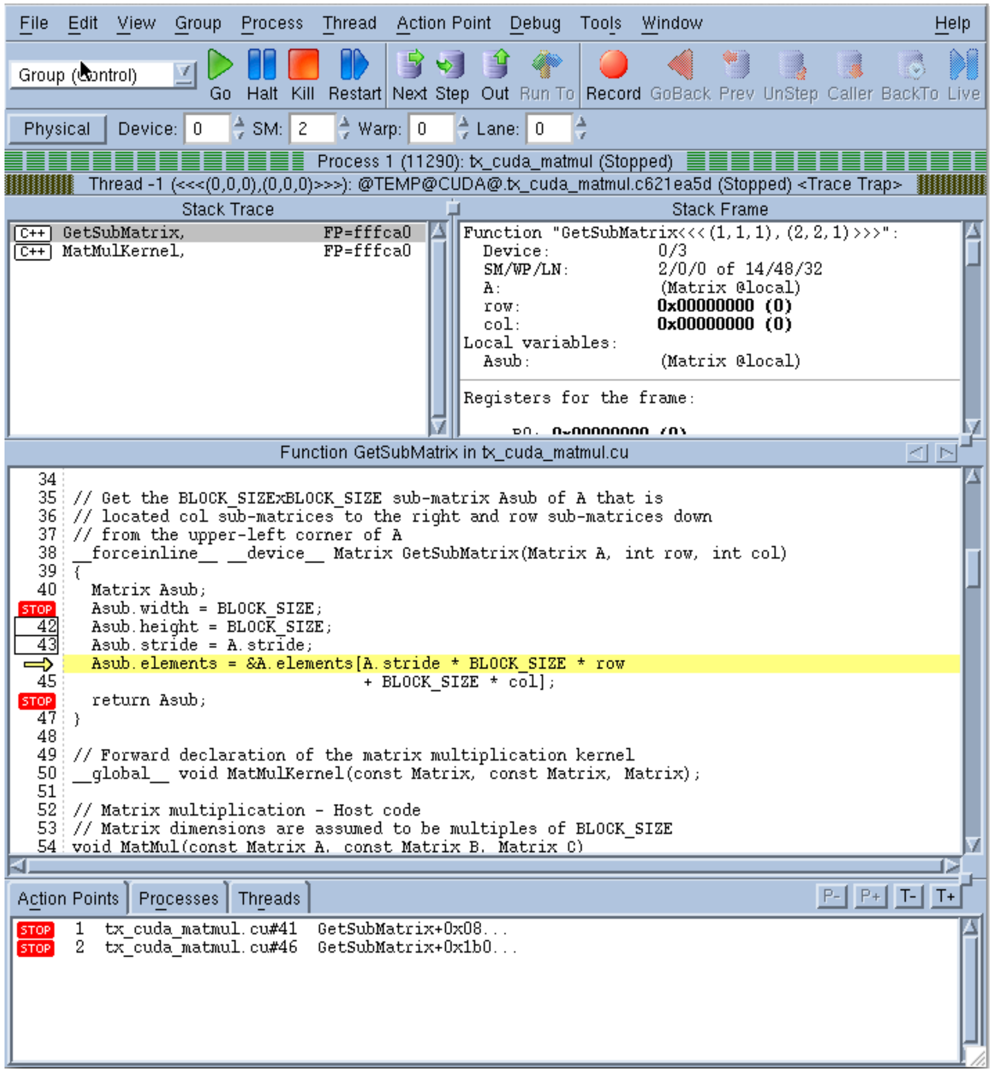
Reports memory access errors

- CUDA memcheck

Multi-Device Support

Can be used with MPI

Starting TotalView


File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
Physical Device: 0 SM: 2 Warp: 0 Lane: 0
Stack Trace
Stack Frame
Action Points Processes Threads
P- P+ T- T+

```
Process 1 (11290): tx_cuda_matmul (Stopped)
Thread -1 (<<<(0,0,0),(0,0,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (Stopped) <Trace Trap>
```

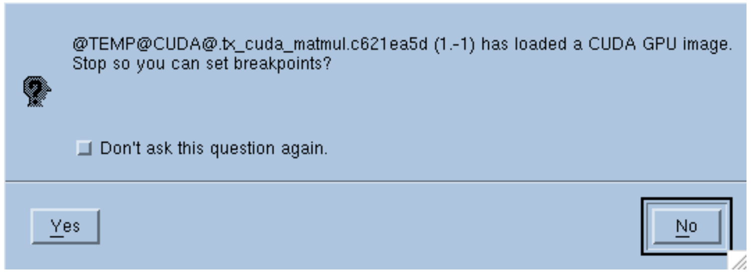
```
C++ GetSubMatrix,        FP=fffca0
C++ MatMulKernel,        FP=fffca0
```

```
Function "GetSubMatrix<<<(1,1,1),(2,2,1)>>>":
  Device:          0/3
  SM/WP/LN:        2/0/0 of 14/48/32
  A:               (Matrix @local)
  row:             0x00000000 (0)
  col:             0x00000000 (0)
Local variables:
  Asub:            (Matrix @local)

Registers for the frame:
```

Function GetSubMatrix in tx_cuda_matmul.cu

```

// Get the BLOCK_SIZExBLOCK_SIZE sub-matrix Asub of A that is
// located col sub-matrices to the right and row sub-matrices down
// from the upper-left corner of A
__forceinline__ __device__ Matrix GetSubMatrix(Matrix A, int row, int col)
{
  Matrix Asub;
  Asub.width = BLOCK_SIZE;
  Asub.height = BLOCK_SIZE;
  Asub.stride = A.stride;
  Asub.elements = &A.elements[A.stride * BLOCK_SIZE * row
                              + BLOCK_SIZE * col];
  return Asub;
}

// Forward declaration of the matrix multiplication kernel
__global__ void MatMulKernel(const Matrix, const Matrix, Matrix);

// Matrix multiplication - Host code
// Matrix dimensions are assumed to be multiples of BLOCK_SIZE
void MatMul(const Matrix A, const Matrix B, Matrix C)
```

```
STOP 1  tx_cuda_matmul.cu#41  GetSubMatrix+0x08...
STOP 2  tx_cuda_matmul.cu#46  GetSubMatrix+0x1b0...
```



@TEMP@CUDA@.tx_cuda_matmul.c621ea5d (1.-1) has loaded a CUDA GPU image. Stop so you can set breakpoints?

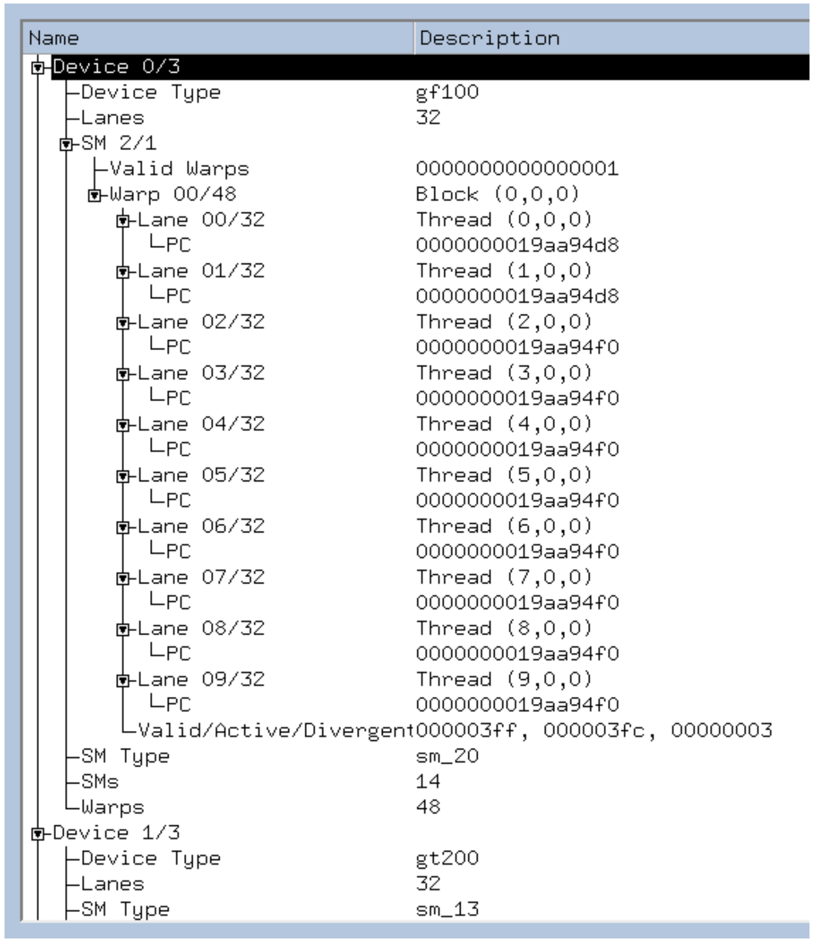
Don't ask this question again.

- Once a new kernel is loaded TotalView provides the option to stop and set breakpoints
- TotalView automatically configures the GUI for CUDA debugging
- Debugging CUDA code is done by using normal TotalView commands and procedures

GPU Device Status Display

Provides the “high-level” view

- Values automatically update as you step through code
- Shows what hardware is in use
- Helps to map between logical and hardware coordinates

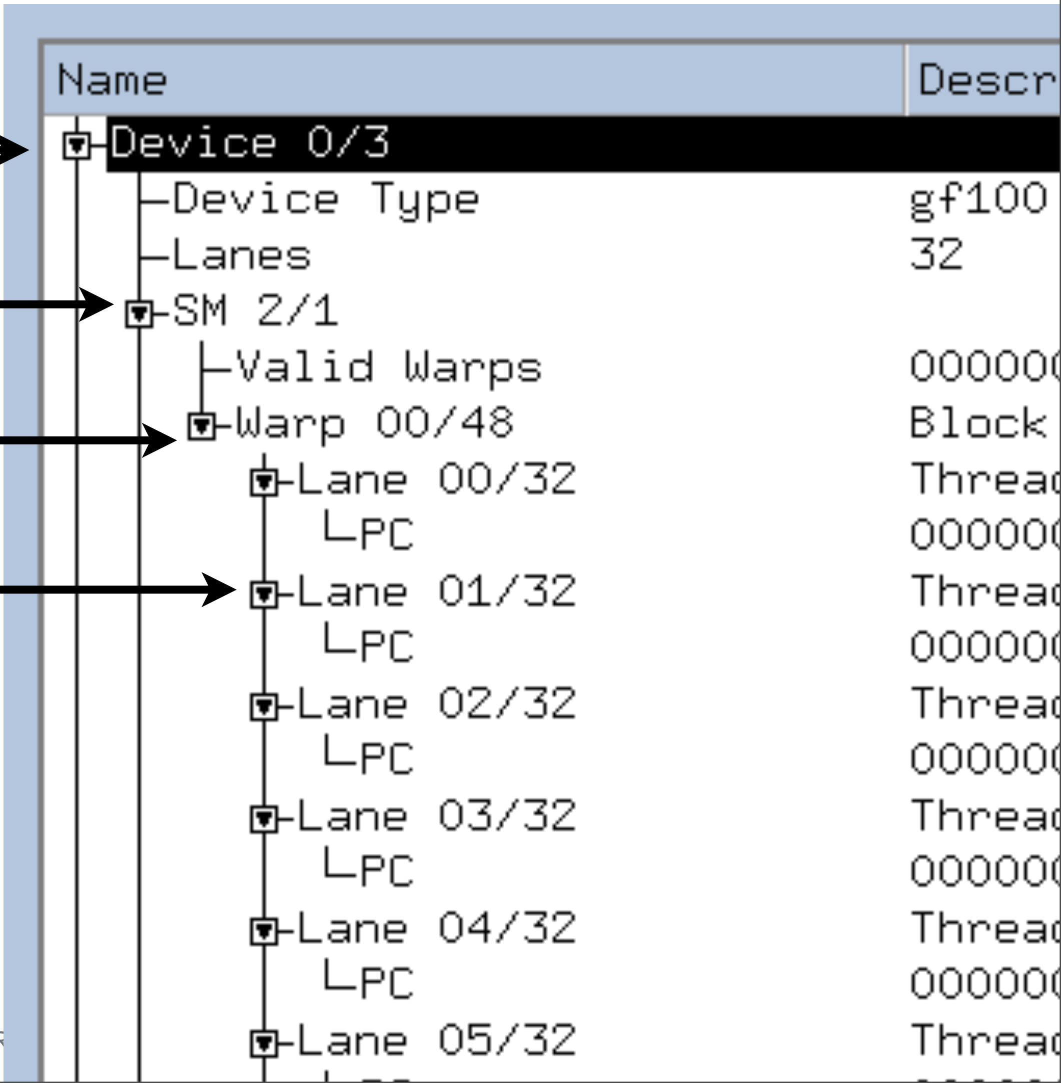


```
Name                                  Description
Device 0/3
 ├Device Type                         gf100
 ├Lanes                               32
 ├SM 2/1
 │ ├Valid Warps                       0000000000000001
 │ ├Warp 00/48                        Block (0,0,0)
 │ │ ├Lane 00/32                      Thread (0,0,0)
 │ │ │ └PC                            0000000019aa94d8
 │ │ ├Lane 01/32                      Thread (1,0,0)
 │ │ │ └PC                            0000000019aa94d8
 │ │ ├Lane 02/32                      Thread (2,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 03/32                      Thread (3,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 04/32                      Thread (4,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 05/32                      Thread (5,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 06/32                      Thread (6,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 07/32                      Thread (7,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 08/32                      Thread (8,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ ├Lane 09/32                      Thread (9,0,0)
 │ │ │ └PC                            0000000019aa94f0
 │ │ └Valid/Active/Divergent000003ff, 000003fc, 00000003
 ├SM Type                             sm_20
 ├SMs                                 14
 └Warps                               48
Device 1/3
 ├Device Type                         gt200
 ├Lanes                               32
 ├SM Type                             sm_13
```

GPU Device Status Display

Provides detailed information for:

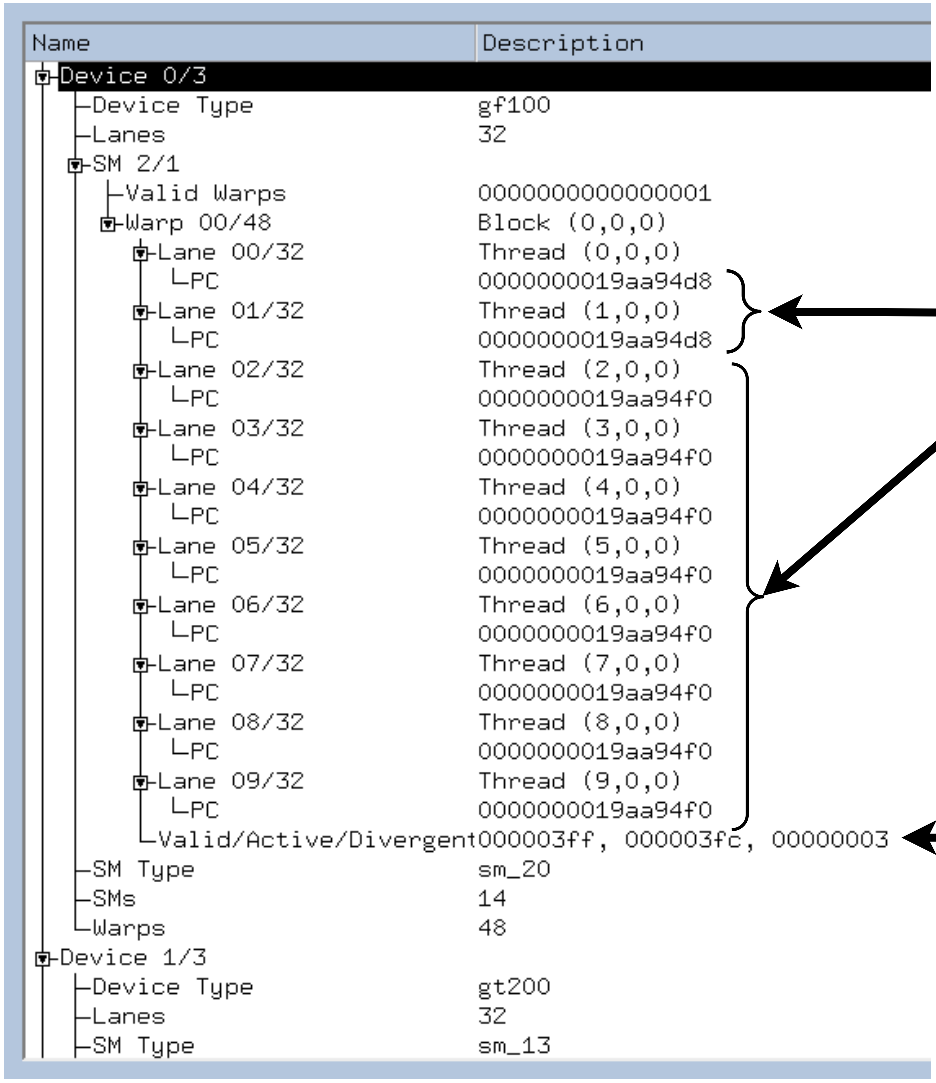
- **Device and Type**
- **SMs**
- **Warps**
- **Lanes with PC**



```
Name                                        Descr
⊟Device 0/3
 ├Device Type                               gf100
 ├Lanes                                     32
 ⊟SM 2/1
  ├Valid Warps                              00000
  ⊟Warp 00/48                               Block
   ⊟Lane 00/32                              Threa
   │ └PC                                    00000
   ⊟Lane 01/32                              Threa
   │ └PC                                    00000
   ⊟Lane 02/32                              Threa
   │ └PC                                    00000
   ⊟Lane 03/32                              Threa
   │ └PC                                    00000
   ⊟Lane 04/32                              Threa
   │ └PC                                    00000
   ⊟Lane 05/32                              Threa
```

Information updates as you step

GPU Device Status Display



```
Name                               Description
Device 0/3
 ├Device Type                      gf100
 ├Lanes                            32
 ├SM 2/1
 │ ├Valid Warps                    0000000000000001
 │ ├Warp 00/48                     Block (0,0,0)
 │ │ ├Lane 00/32                   Thread (0,0,0)
 │ │ │ └PC                         0000000019aa94d8
 │ │ ├Lane 01/32                   Thread (1,0,0)
 │ │ │ └PC                         0000000019aa94d8
 │ │ ├Lane 02/32                   Thread (2,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 03/32                   Thread (3,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 04/32                   Thread (4,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 05/32                   Thread (5,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 06/32                   Thread (6,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 07/32                   Thread (7,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 08/32                   Thread (8,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ ├Lane 09/32                   Thread (9,0,0)
 │ │ │ └PC                         0000000019aa94f0
 │ │ └Valid/Active/Divergent000003ff, 000003fc, 00000003
 ├SM Type                          sm_20
 ├SMs                              14
 └Warps                            48
Device 1/3
 ├Device Type                      gt200
 ├Lanes                            32
 ├SM Type                          sm_13
```

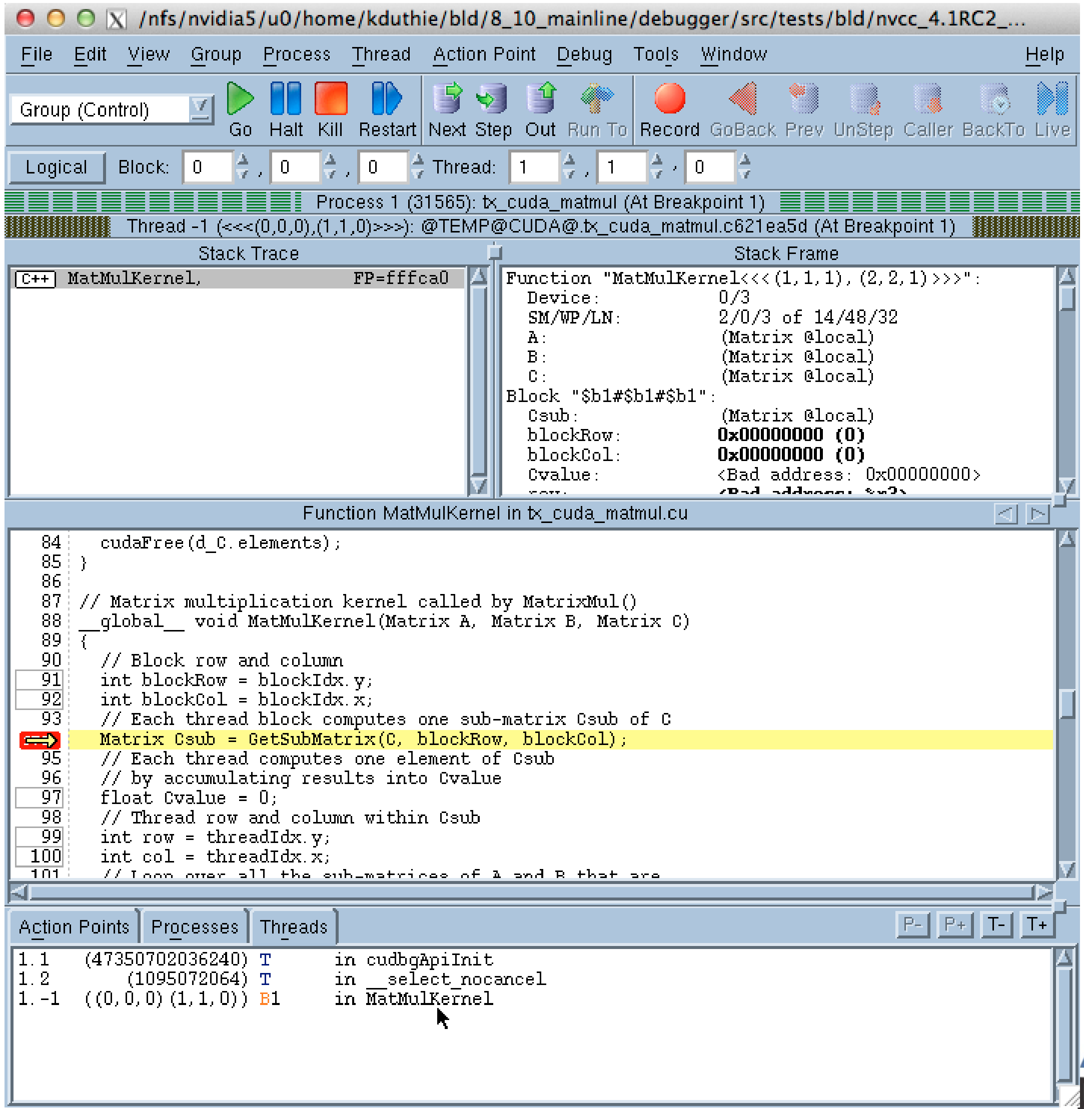
It also provides information for divergent GPU threads

Different PC for two groups of Lanes

State of Lanes inside the Warp

Debugging CUDA

Information on GPU execution, location and data is readily available.
... the same as it is for Linux processes and threads.


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debugger/src/tests/bld/nvcc_4.1RC2_...
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
Logical

Block: 0, 0, 0 Thread: 1, 1, 0

Process 1 (31565): tx_cuda_matmul (At Breakpoint 1)

Thread -1 (<<<(0,0,0),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (At Breakpoint 1)

Stack Trace

```
C++  MatMulKernel,                   FP=fffca0
```

Stack Frame

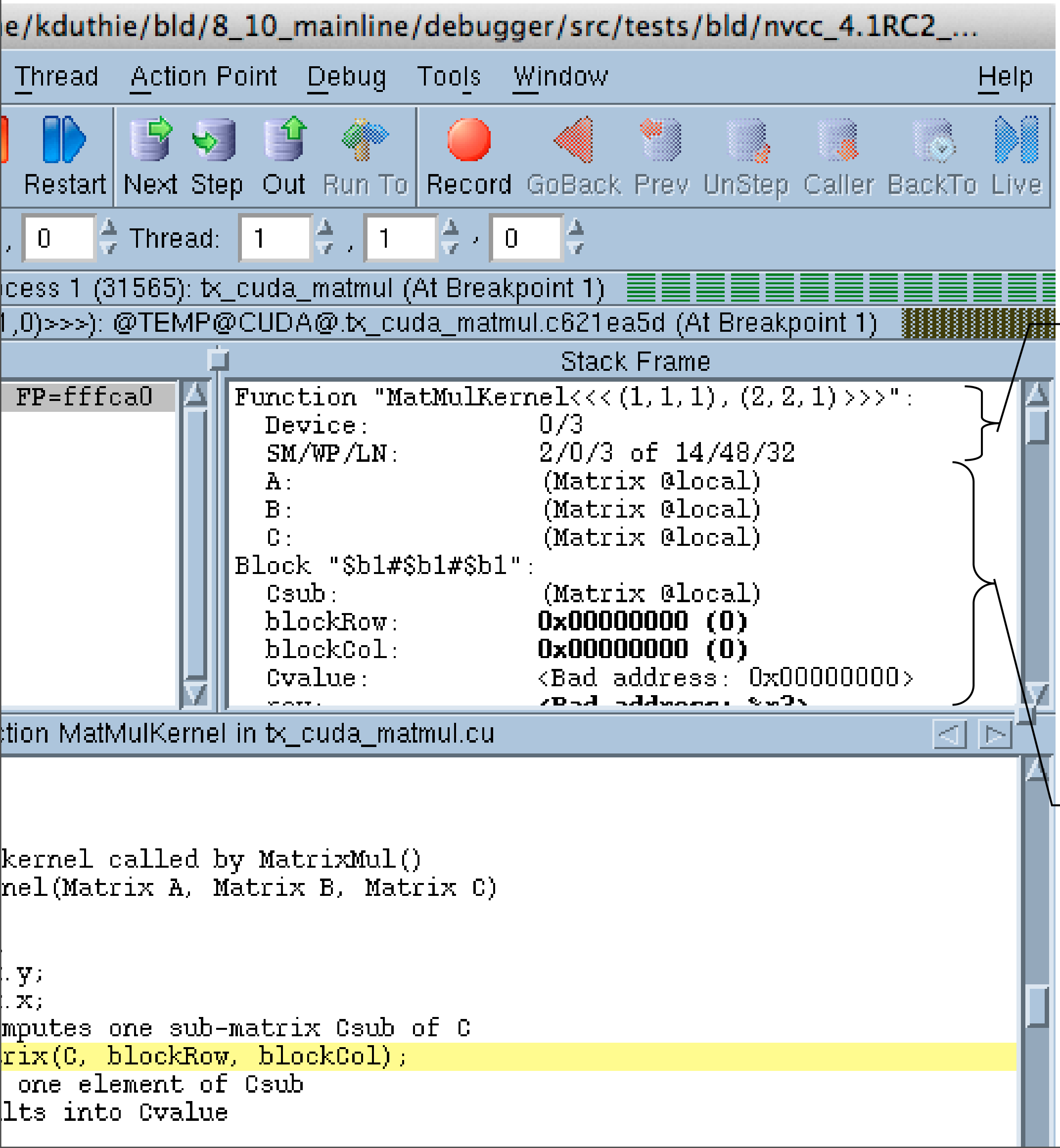
```
Function "MatMulKernel<<<(1,1,1),(2,2,1)>>>":
  Device:                0/3
  SM/WP/LN:              2/0/3 of 14/48/32
  A:                     (Matrix @local)
  B:                     (Matrix @local)
  C:                     (Matrix @local)
Block "$b1#$b1#$b1":
  Csub:                  (Matrix @local)
  blockRow:              0x00000000 (0)
  blockCol:              0x00000000 (0)
  Cvalue:                <Bad address: 0x00000000>
```

Function MatMulKernel in tx_cuda_matmul.cu

```
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by MatrixMul()
__global__ void MatMulKernel(Matrix A, Matrix B, Matrix C)
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix Csub of C
  Matrix Csub = GetSubMatrix(C, blockRow, blockCol);
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
```

```
1.1    (47350702036240) T      in cudbgApiInit
1.2        (1095072064) T      in __select_nocancel
1.-1   ((0,0,0)(1,1,0)) B1     in MatMulKernel
```

Debugging CUDA


e/kduthie/bld/8_10_mainline/debugger/src/tests/bld/nvcc_4.1RC2_...
Thread Action Point Debug Tools Window Help
Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
, 0 Thread: 1 , 1 , 0
Stack Frame
FP=fffca0
tion MatMulKernel in tx_cuda_matmul.cu

```
cess 1 (31565): tx_cuda_matmul (At Breakpoint 1)
1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (At Breakpoint 1)
```

```
Function "MatMulKernel<<<(1,1,1),(2,2,1)>>>":
  Device:              0/3
  SM/WP/LN:            2/0/3 of 14/48/32
  A:                   (Matrix @local)
  B:                   (Matrix @local)
  C:                   (Matrix @local)
Block "$b1#$b1#$b1":
  Csub:                (Matrix @local)
  blockRow:            0x00000000 (0)
  blockCol:            0x00000000 (0)
  Cvalue:              <Bad address: 0x00000000>
```

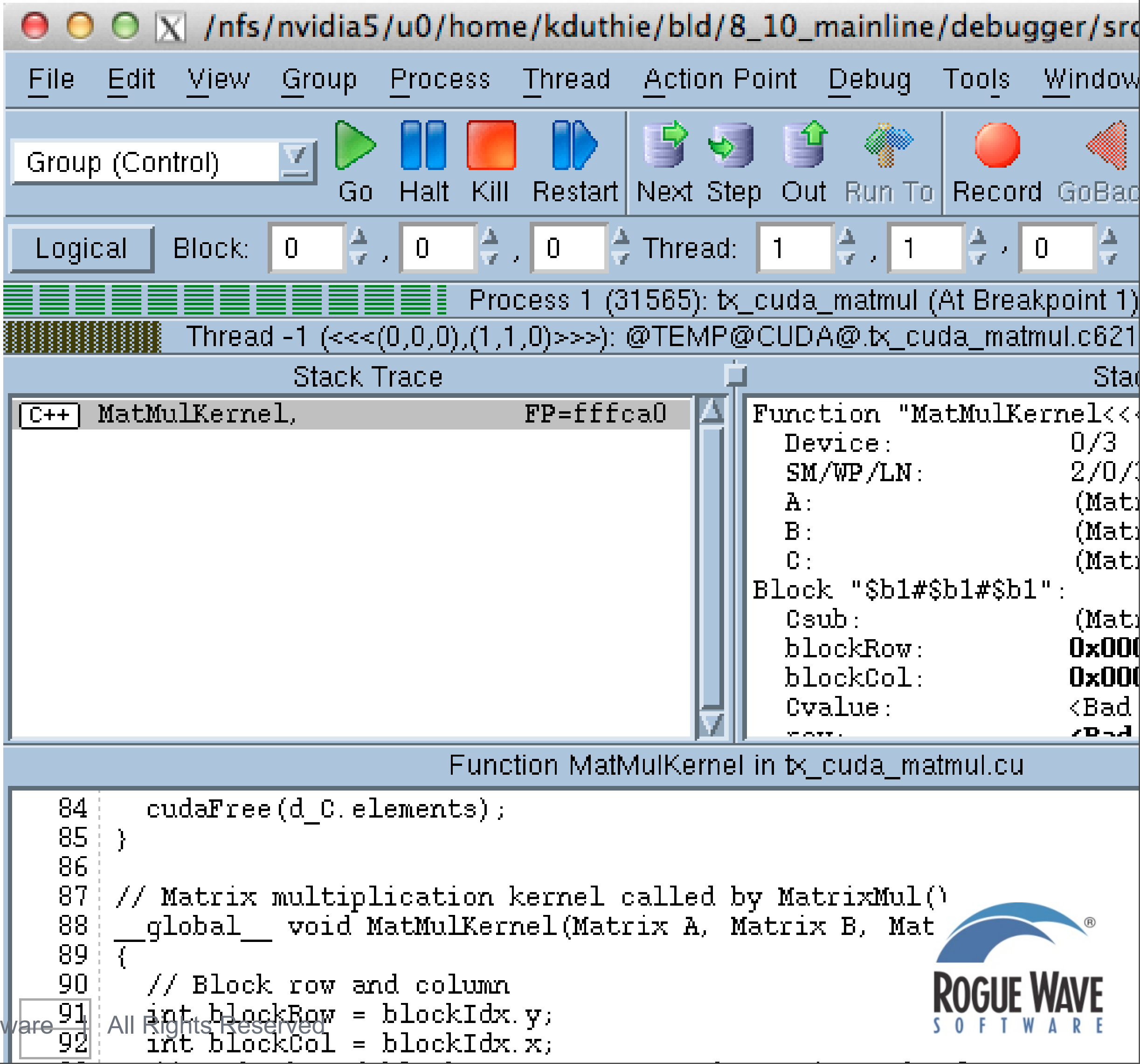
```
kernel called by MatrixMul()
nel(Matrix A, Matrix B, Matrix C)

.y;
.x;
mputes one sub-matrix Csub of C
rix(C, blockRow, blockCol);
 one element of Csub
lts into Cvalue
```

CUDA grid and block dimensions, lanes/ warp, warps/SM,

Parameter, register, local and shared variables

Debugging CUDA

GPU focus thread logical coordinates in the header...


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debugger/src
File Edit View Group Process Thread Action Point Debug Tools Window
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBac
Logical Block: 0 , 0 , 0 Thread: 1 , 1 , 0
Stack Trace
Stac

```
Process 1 (31565): tx_cuda_matmul (At Breakpoint 1)
Thread -1 (<<<(0,0,0),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621
```

```
C++  MatMulKernel,                FP=fffca0
```

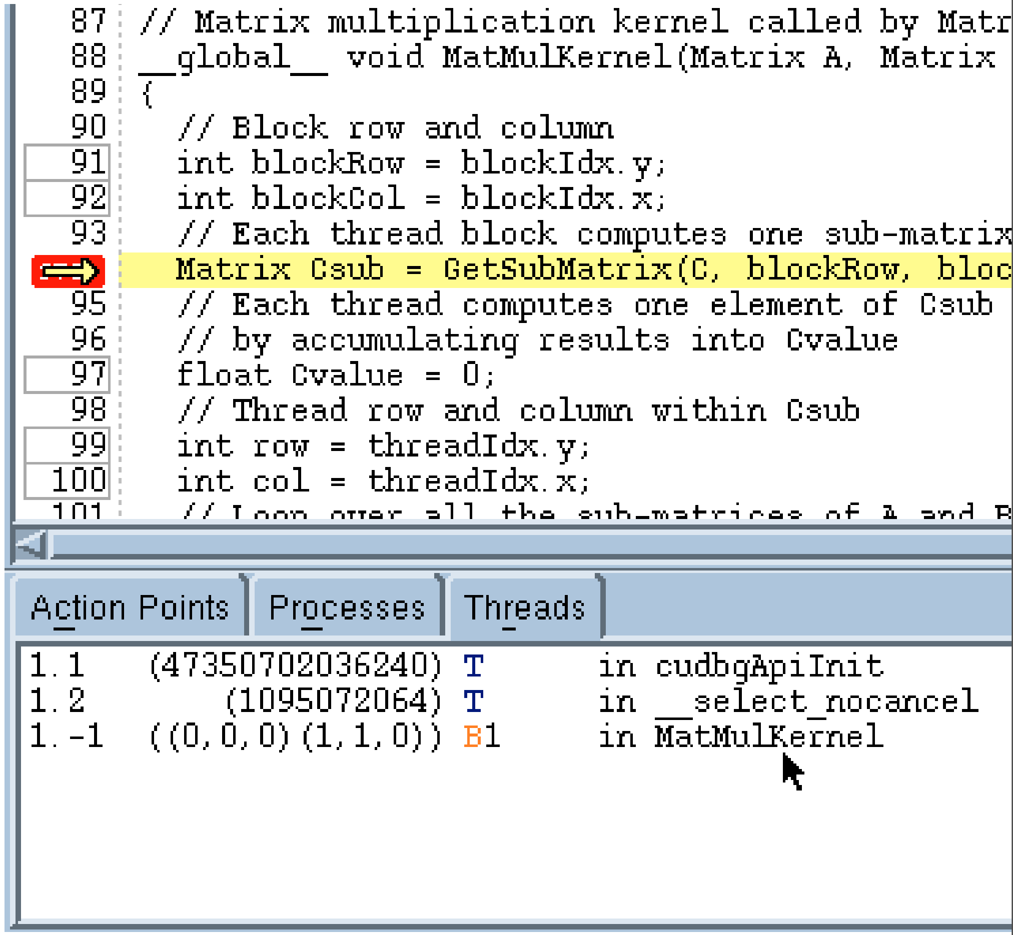
```
Function "MatMulKernel<<
  Device:           0/3
  SM/WP/LN:         2/0/
  A:                (Mat
  B:                (Mat
  C:                (Mat
Block "$b1#$b1#$b1":
  Csub:             (Mat
  blockRow:         0x00
  blockCol:         0x00
  Cvalue:           <Bad
```

Function MatMulKernel in tx_cuda_matmul.cu

```
   cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by MatrixMul()
__global__ void MatMulKernel(Matrix A, Matrix B, Mat
{
   // Block row and column
   int blockRow = blockIdx.y;
   int blockCol = blockIdx.x;
```

Debugging CUDA

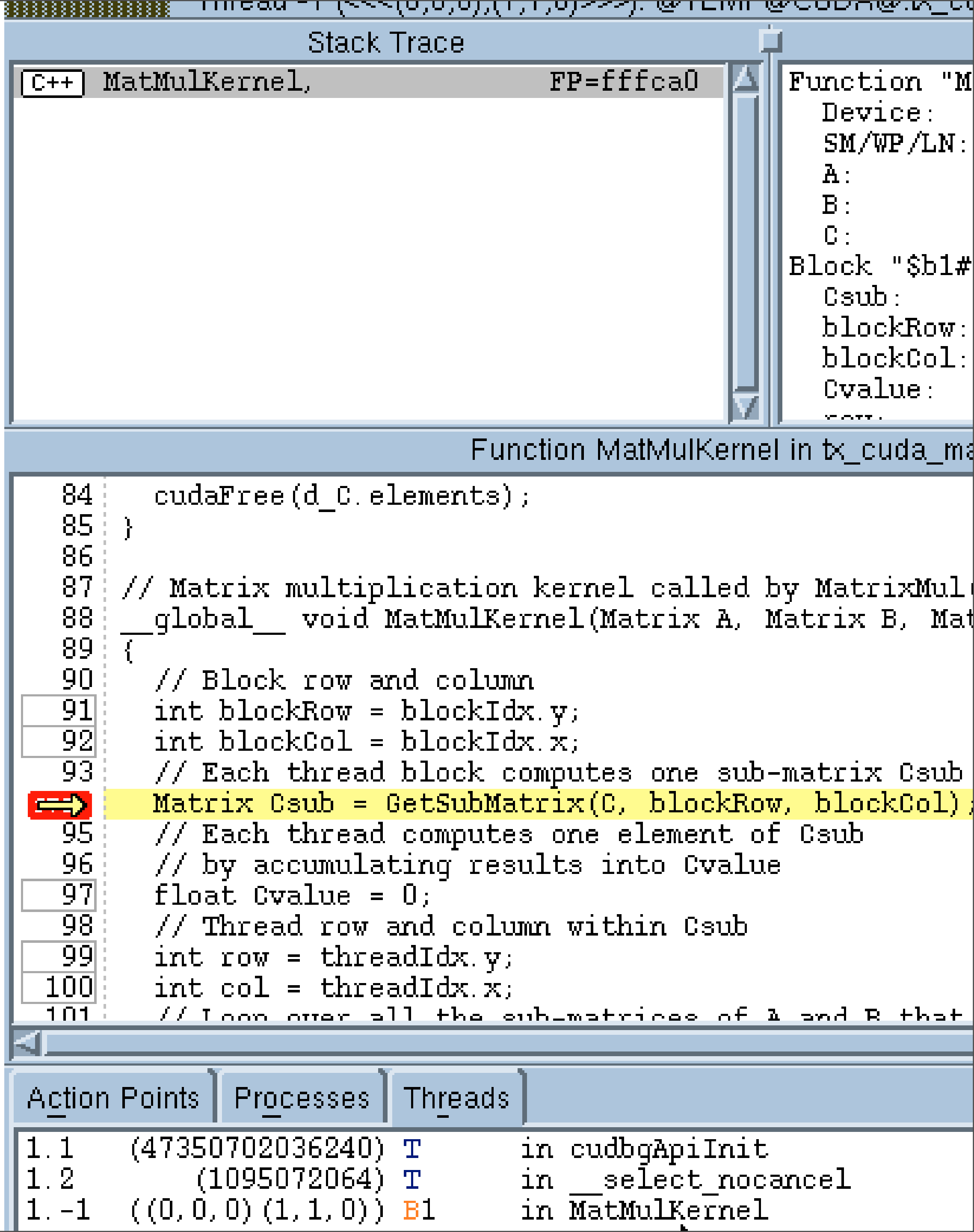
... as well as in the Process Window



```
// Matrix multiplication kernel called by Matr
__global__ void MatMulKernel(Matrix A, Matrix
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix
  Matrix Csub = GetSubMatrix(C, blockRow, bloc
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
  // Loop over all the sub-matrices of A and B
```

```
1.1    (47350702036240) T     in cudbgApiInit
1.2        (1095072064) T     in __select_nocancel
1.-1   ((0,0,0)(1,1,0)) B1    in MatMulKernel
```

Debugging CUDA



Stack Trace

```
C++  MatMulKernel,                FP=fffca0
```

```
Function "M
  Device:
  SM/WP/LN:
  A:
  B:
  C:
Block "$b1#
  Csub:
  blockRow:
  blockCol:
  Cvalue:
```

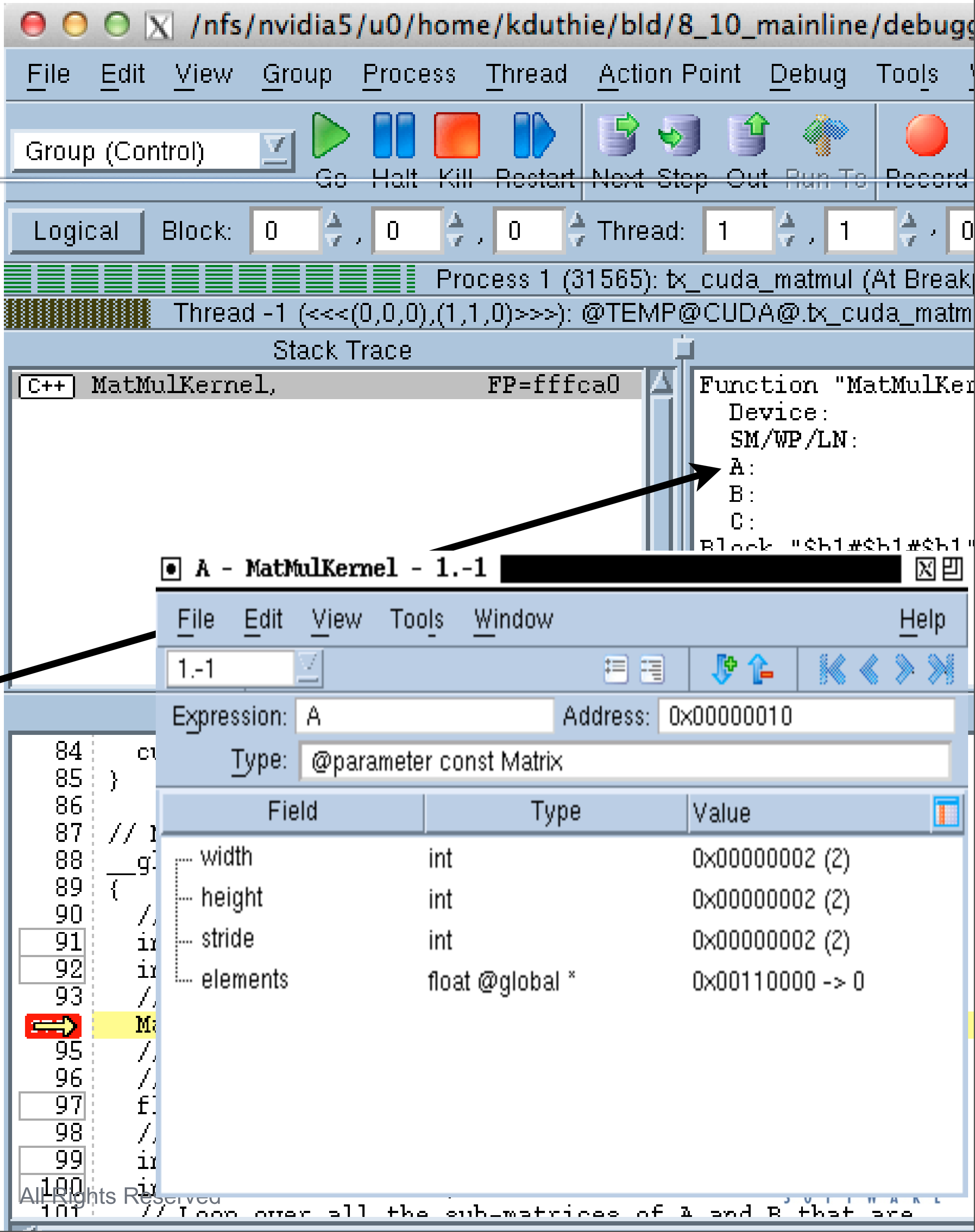
Function MatMulKernel in tx_cuda_ma

```
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by MatrixMul
__global__ void MatMulKernel(Matrix A, Matrix B, Mat
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix Csub
  Matrix Csub = GetSubMatrix(C, blockRow, blockCol);
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
```

```
1.1     (47350702036240) T      in cudbgApiInit
1.2          (1095072064) T      in __select_nocancel
1.-1   ((0,0,0)(1,1,0)) B1      in MatMulKernel
```

PC arrow shows the Program Counter for the warp

Debugging CUDA


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debug
File Edit View Group Process Thread Action Point Debug Tools
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record
Logical Block: 0 , 0 , 0 Thread: 1 , 1 , 0
Process 1 (31565): tx_cuda_matmul (At Break
Thread -1 (<<<(0,0,0),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matm
Stack Trace
C++ MatMulKernel, FP=fffca0
A - MatMulKernel - 1.-1
File Edit View Tools Window Help
1.-1

```
Function "MatMulKer
  Device:
  SM/WP/LN:
  A:
  B:
  C:
Block "$b1#$b1#$b1"
```

Expression: A   Address: 0x00000010

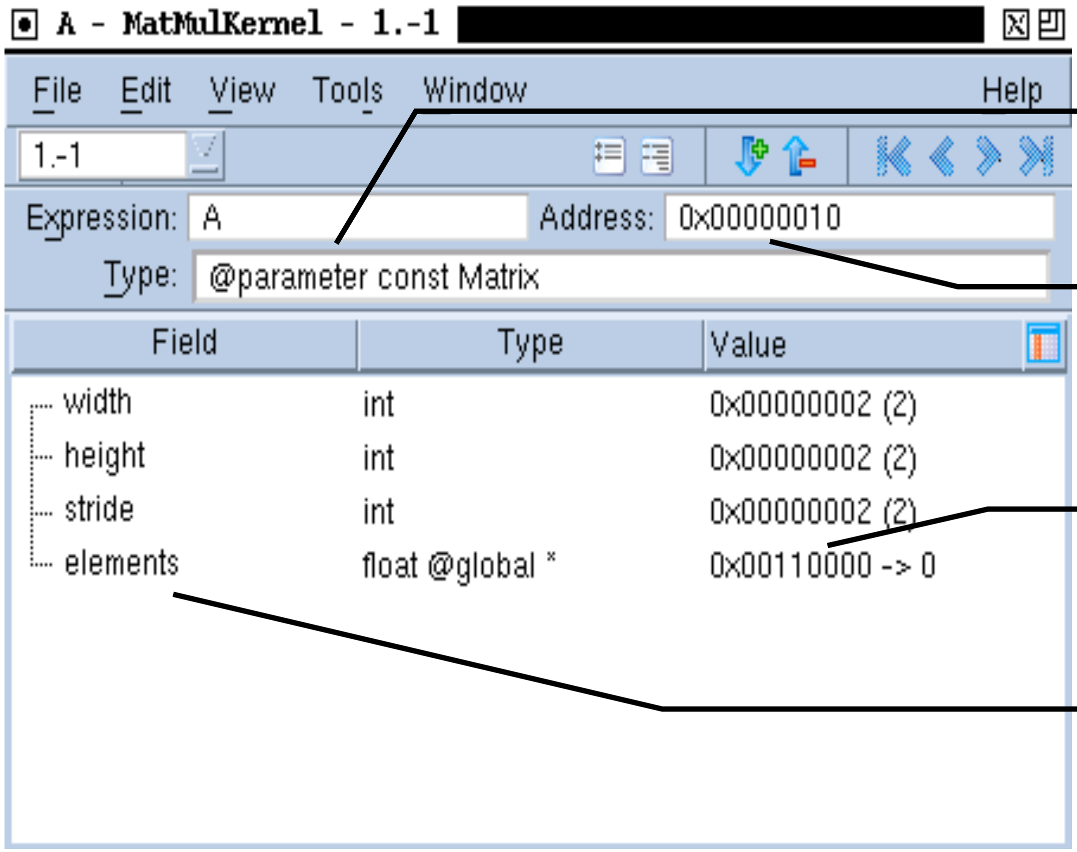
Type: @parameter const Matrix

Field	Type	Value
width	int	0x00000002 (2)
height	int	0x00000002 (2)
stride	int	0x00000002 (2)
elements	float @global *	0x00110000 -> 0

```
84   cu
85 }
86
87 // I
88 __gl
89 {
90   //
91   in
92   in
93   //
94   Ma
95   //
96   //
97   fl
98   //
99   in
100  in
101  // Loop over all the sub-matrices of A and B that are
```

Dive on any variable name to open a variable window

Debugging CUDA



Expression: A   Address: 0x00000010

Type: @parameter const Matrix

Field	Type	Value
width	int	0x00000002 (2)
height	int	0x00000002 (2)
stride	int	0x00000002 (2)
elements	float @global *	0x00110000 -> 0

"@parameter" type qualifier indicates that variable "A" is in parameter storage

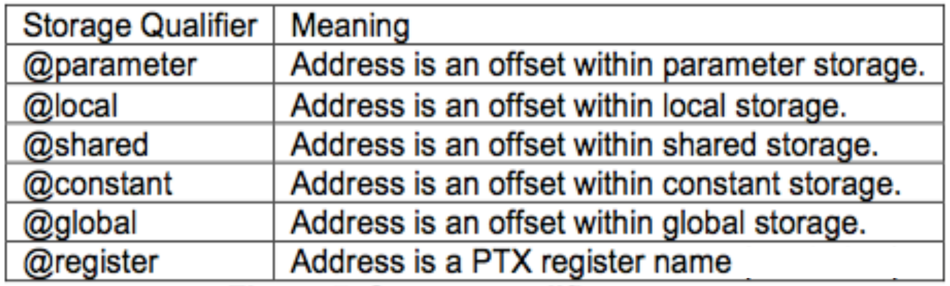
Address 0x10 is an offset within parameter storage

Pointer value 0x110000 is an offset within global storage

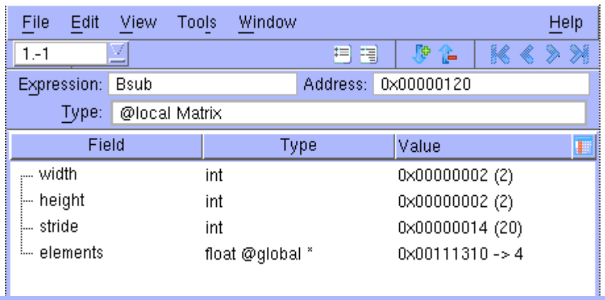
"elements" is a pointer to a float in global storage

Storage Qualifiers

- Denotes location in hierarchical memory
  - Part of the type – using “@” notation
  - Each memory space has a separate address space so 0x00001234 could refer to several places

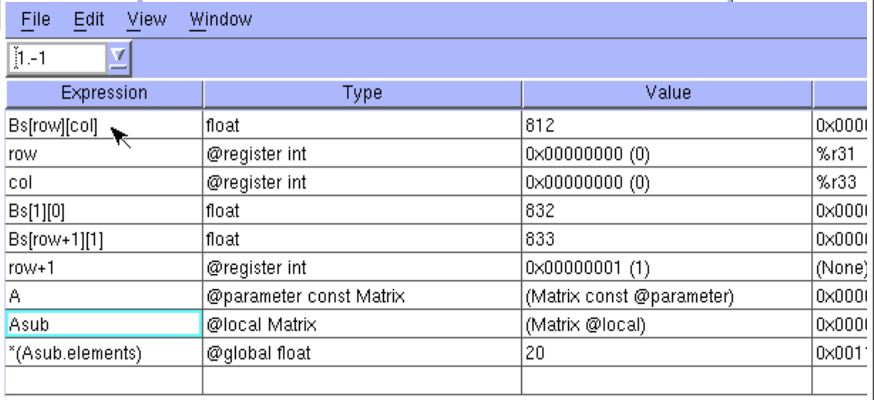


Storage Qualifier	Meaning
@parameter	Address is an offset within parameter storage.
@local	Address is an offset within local storage.
@shared	Address is an offset within shared storage.
@constant	Address is an offset within constant storage.
@global	Address is an offset within global storage.
@register	Address is a PTX register name



Expression: Bsub Address: 0x00000120
Type: @local Matrix

Field	Type	Value
width	int	0x00000002 (2)
height	int	0x00000002 (2)
stride	int	0x00000014 (20)
elements	float @global *	0x00111310 -> 4

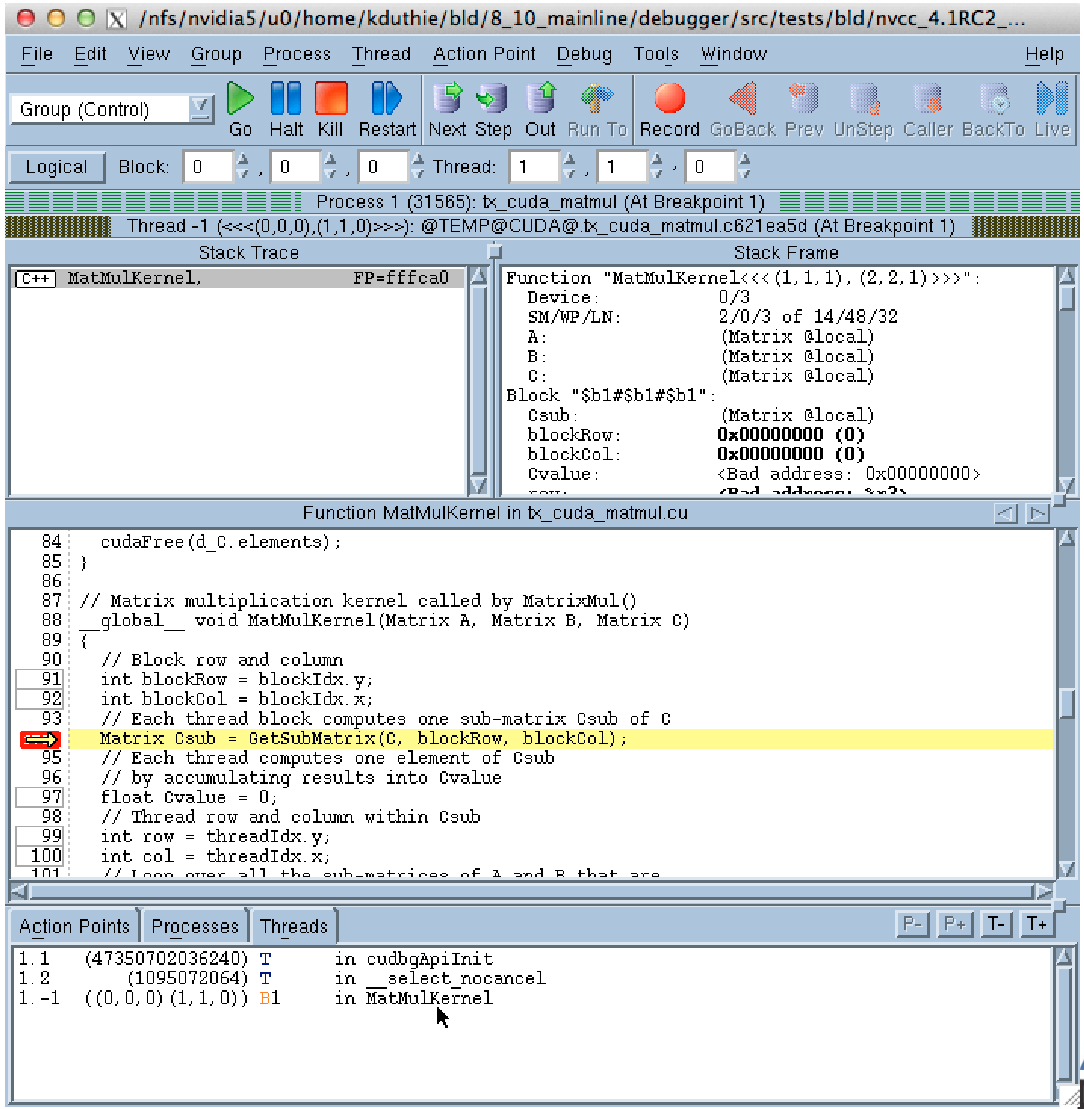


Expression	Type	Value	
Bs[row][col]	float	812	0x000
row	@register int	0x00000000 (0)	%r31
col	@register int	0x00000000 (0)	%r33
Bs[1][0]	float	832	0x000
Bs[row+1][1]	float	833	0x000
row+1	@register int	0x00000001 (1)	(None)
A	@parameter const Matrix	(Matrix const @parameter)	0x000
Asub	@local Matrix	(Matrix @local)	0x000
*(Asub.elements)	@global float	20	0x001

- Used throughout expression system
  - You can cast to switch between different spaces

Debugging CUDA - Navigation

Navigate through your CUDA code in the Process Window as you wish...
Using either of two coordinate systems:


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debugger/src/tests/bld/nvcc_4.1RC2_...
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
Logical Block: 0 , 0 , 0 Thread: 1 , 1 , 0

Process 1 (31565): tx_cuda_matmul (At Breakpoint 1)
Thread -1 (<<<(0,0,0),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (At Breakpoint 1)

Stack Trace

```
C++ MatMulKernel,                FP=fffca0
```

Stack Frame

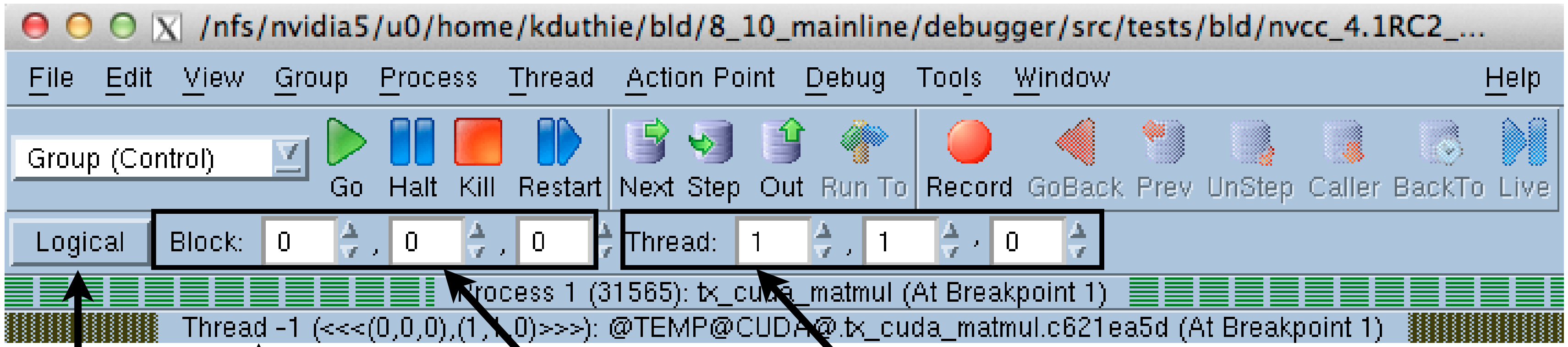
```
Function "MatMulKernel<<<(1,1,1),(2,2,1)>>>":
  Device:             0/3
  SM/WP/LN:           2/0/3 of 14/48/32
  A:                  (Matrix @local)
  B:                  (Matrix @local)
  C:                  (Matrix @local)
Block "$b1#$b1#$b1":
  Csub:               (Matrix @local)
  blockRow:           0x00000000 (0)
  blockCol:           0x00000000 (0)
  Cvalue:             <Bad address: 0x00000000>
```

Function MatMulKernel in tx_cuda_matmul.cu

```
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by MatrixMul()
__global__ void MatMulKernel(Matrix A, Matrix B, Matrix C)
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matrix Csub of C
  Matrix Csub = GetSubMatrix(C, blockRow, blockCol);
  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
```

```
1.1    (47350702036240) T    in cudbgApiInit
1.2         (1095072064) T    in __select_nocancel
1.-1   ((0,0,0) (1,1,0)) B1   in MatMulKernel
```

Debugging CUDA - Navigation


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debugger/src/tests/bld/nvcc_4.1RC2_...
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
Logical
Block: 0 , 0 , 0
Thread: 1 , 1 , 0

Process 1 (31565): tx_cuda_matmul (At Breakpoint 1)

Thread -1 (<<<(0,0,0),(1,1,0)>>>): @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (At Breakpoint 1)

CUDA GPU threads have a negative TotalView thread ID

Block (x,y,z)

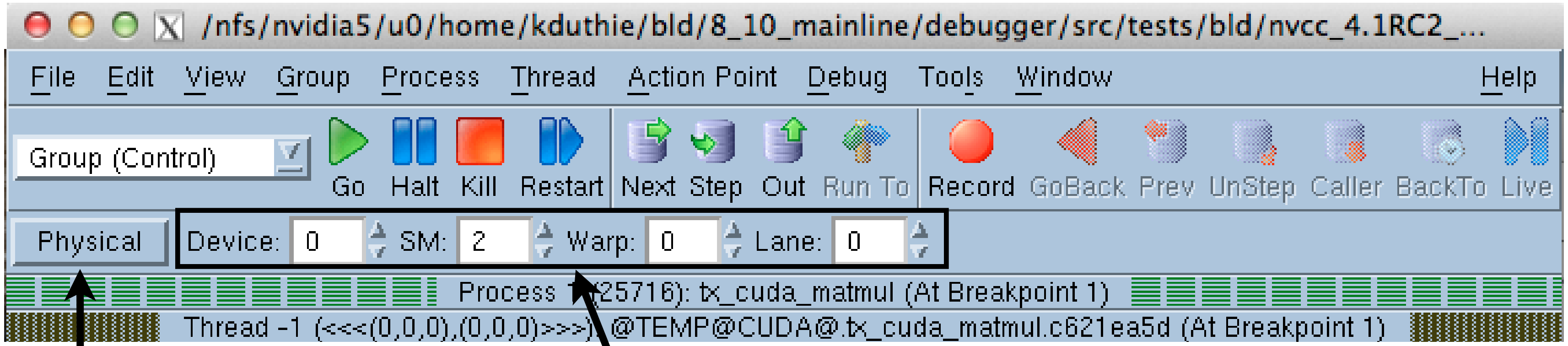
Thread (x,y,z)

User-controlled "spinboxes" allow selection and display of any part of your GPU execution

GPU focus thread selector for changing the logical block and thread indexes of the CUDA thread.

- Logical: 2 or 3D Grid of Blocks, 3D Thread Within Grid

Debugging CUDA - Navigation


/nfs/nvidia5/u0/home/kduthie/bld/8_10_mainline/debugger/src/tests/bld/nvcc_4.1RC2_...
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live
Physical

Device: 0 SM: 2 Warp: 0 Lane: 0

Process 1 (25716): tx_cuda_matmul (At Breakpoint 1)

Thread -1 (<<<(0,0,0),(0,0,0)>>>) @TEMP@CUDA@.tx_cuda_matmul.c621ea5d (At Breakpoint 1)

Device, SM, Warp, and Lane

User-controlled "spinboxes" allow selection and display of any part of your GPU execution

GPU focus selector for changing physical indexes of the CUDA thread.

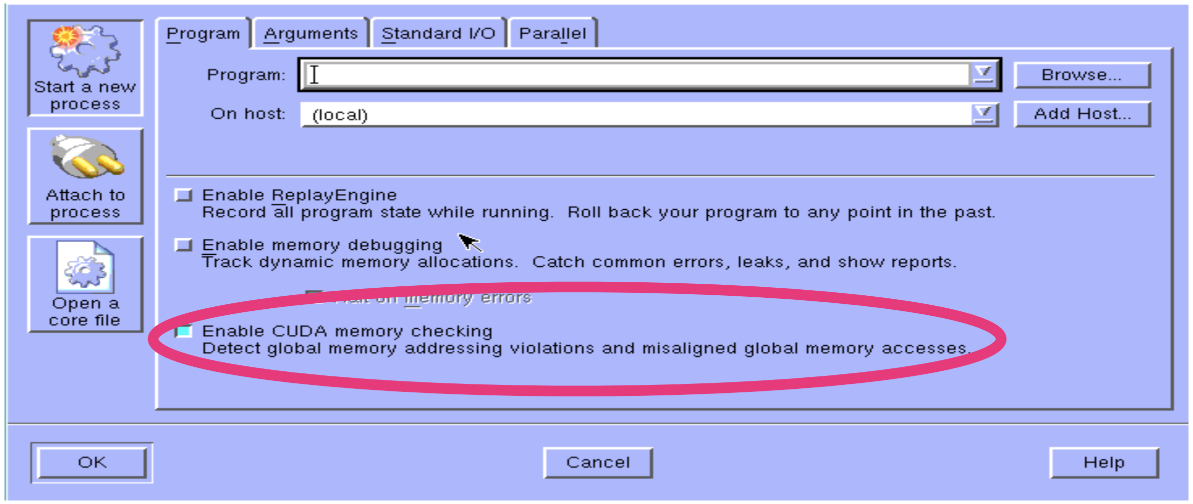
- **Physical: Device, SM, Warp, Lane**

Executing GPU Code - Threads and Warps

- Single-step operation advances all of the GPU hardware threads in the *same* warp
- To advance the execution of more than one warp:
  - set a breakpoint and continue the process, or
  - select a line number in the source pane and select “Run To”.
- Warps advance synchronously
  - Warps share a PC
- Single stepping
  - Advances the warp containing the focus thread
  - Stepping over a __syncthreads() call advances all the relevant threads
- Continue and runto
  - Continues more than just the warp
- Halt
  - Stops all the host and device threads

CUDA Segmentation Faults

- **TotalView displays segmentation faults as expected**
  - **Enable CUDA memory checking in New Program dialog window**



Program:

On host: (local)

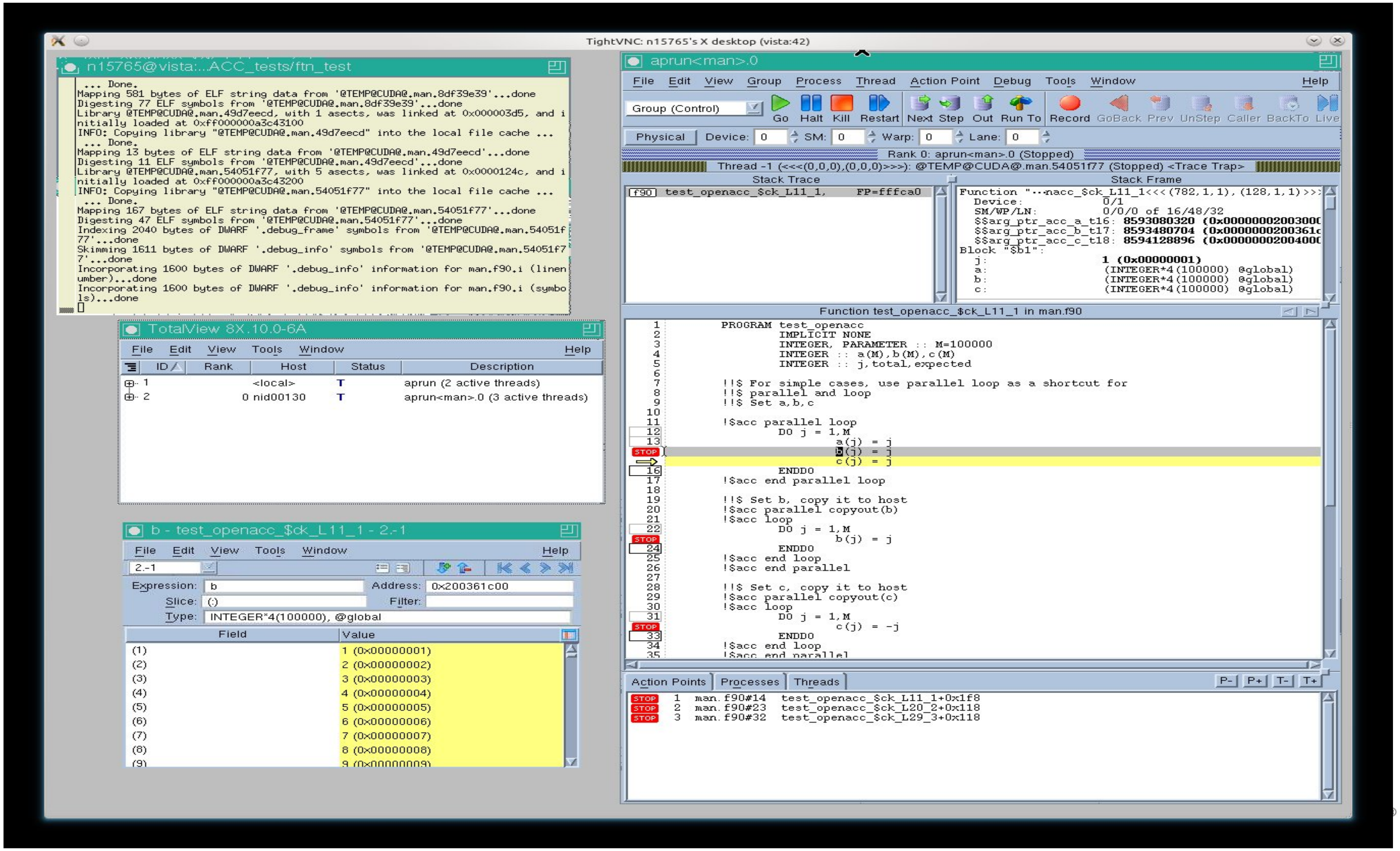
☐ Enable ReplayEngine
Record all program state while running. Roll back your program to any point in the past.

☐ Enable memory debugging
Track dynamic memory allocations. Catch common errors, leaks, and show reports.

☐ Halt on memory errors

☐ Enable CUDA memory checking
Detect global memory addressing violations and misaligned global memory accesses.

OpenACC



TightVNC: n15765's X desktop (vista:42)

n15765@vista:...ACC_tests/ftn_test

```
 ... Done.
Mapping 581 bytes of ELF string data from '@TEMP@CUDA@.man.8df39e39'...done
Digesting 77 ELF symbols from '@TEMP@CUDA@.man.8df39e39'...done
Library @TEMP@CUDA@.man.49d7eecd, with 1 asects, was linked at 0x000003d5, and i
nitially loaded at 0xff000000a3c43100
INFO: Copying library "@TEMP@CUDA@.man.49d7eecd" into the local file cache ...
 ... Done.
Mapping 13 bytes of ELF string data from '@TEMP@CUDA@.man.49d7eecd'...done
Digesting 11 ELF symbols from '@TEMP@CUDA@.man.49d7eecd'...done
Library @TEMP@CUDA@.man.54051f77, with 5 asects, was linked at 0x0000124c, and i
nitially loaded at 0xff000000a3c43200
INFO: Copying library "@TEMP@CUDA@.man.54051f77" into the local file cache ...
 ... Done.
Mapping 167 bytes of ELF string data from '@TEMP@CUDA@.man.54051f77'...done
Digesting 47 ELF symbols from '@TEMP@CUDA@.man.54051f77'...done
Indexing 2040 bytes of DWARF '.debug_frame' symbols from '@TEMP@CUDA@.man.54051f
77'...done
Skimming 1611 bytes of DWARF '.debug_info' symbols from '@TEMP@CUDA@.man.54051f7
7'...done
Incorporating 1600 bytes of DWARF '.debug_info' information for man.f90.i (linen
umber)...done
Incorporating 1600 bytes of DWARF '.debug_info' information for man.f90.i (symbo
ls)...done
```

TotalView 8X.10.0-6A

ID	Rank	Host	Status	Description
1		<local>	T	aprun (2 active threads)
2	0	nid00130	T	aprun<man>.0 (3 active threads)

b - test_openacc_$ck_L11_1 - 2.-1

2.-1

Expression: b Address: 0x200361c00
Slice: (:) Filter:
Type: INTEGER*4(100000), @global

Field	Value
(1)	1 (0x00000001)
(2)	2 (0x00000002)
(3)	3 (0x00000003)
(4)	4 (0x00000004)
(5)	5 (0x00000005)
(6)	6 (0x00000006)
(7)	7 (0x00000007)
(8)	8 (0x00000008)
(9)	9 (0x00000009)

aprun<man>.0

Group (Control)

Physical Device: 0 SM: 0 Warp: 0 Lane: 0

Rank 0: aprun<man>.0 (Stopped)

Thread -1 (<<<(0,0,0),(0,0,0)>>>): @TEMP@CUDA@.man.54051f77 (Stopped) <Trace Trap>

Stack Trace

```
f90  test_openacc_$ck_L11_1,      FP=fffca0
```

Stack Frame

```
Function "…nacc_$ck_L11_1<<<(782,1,1),(128,1,1)>>>
  Device:                 0/1
  SM/WP/LN:               0/0/0 of 16/48/32
  $$arg_ptr_acc_a_t16:  8593080320  (0x000000020030000
  $$arg_ptr_acc_b_t17:  8593480704  (0x00000000200361c
  $$arg_ptr_acc_c_t18:  8594128896  (0x000000020040000
Block "$b1":
  j:                      1  (0x00000001)
  a:                      (INTEGER*4(100000)  @global)
  b:                      (INTEGER*4(100000)  @global)
  c:                      (INTEGER*4(100000)  @global)
```

Function test_openacc_$ck_L11_1 in man.f90

```
        PROGRAM test_openacc
                IMPLICIT NONE
                INTEGER, PARAMETER :: M=100000
                INTEGER :: a(M),b(M),c(M)
                INTEGER :: j,total,expected

        !!$ For simple cases, use parallel loop as a shortcut for
        !!$ parallel and loop
        !!$ Set a,b,c

        !$acc parallel loop
                DO j = 1,M
                        a(j) = j
STOP                    b(j) = j
                        c(j) = j
                ENDDO
        !$acc end parallel loop

        !!$ Set b, copy it to host
        !$acc parallel copyout(b)
        !$acc loop
                DO j = 1,M
STOP                    b(j) = j
                ENDDO
        !$acc end loop
        !$acc end parallel

        !!$ Set c, copy it to host
        !$acc parallel copyout(c)
        !$acc loop
                DO j = 1,M
STOP                    c(j) = -j
                ENDDO
        !$acc end loop
        !$acc end parallel
```

Action Points | Processes | Threads

```
STOP  1  man.f90#14  test_openacc_$ck_L11_1+0x1f8
STOP  2  man.f90#23  test_openacc_$ck_L20_2+0x118
STOP  3  man.f90#32  test_openacc_$ck_L29_3+0x118
```

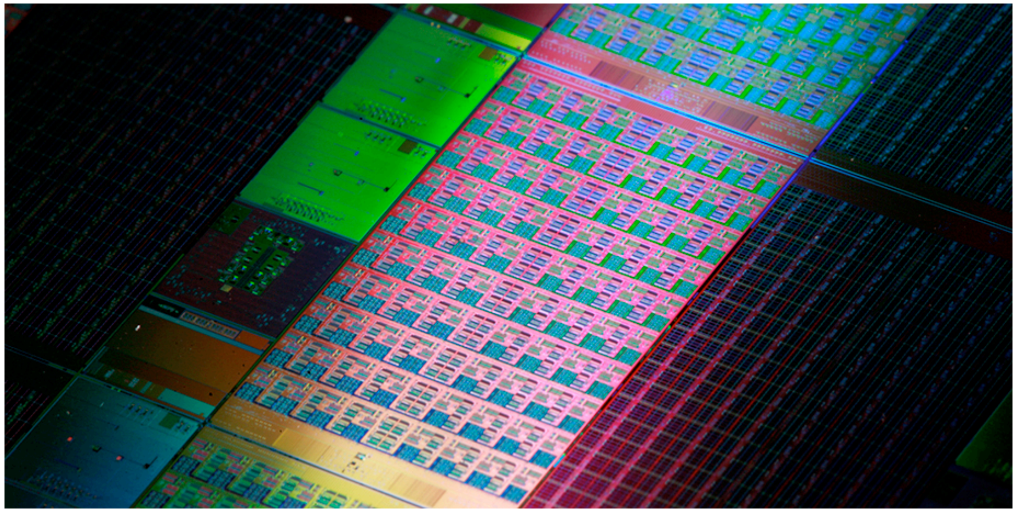
ROGUE WAVE SOFTWARE

What's New

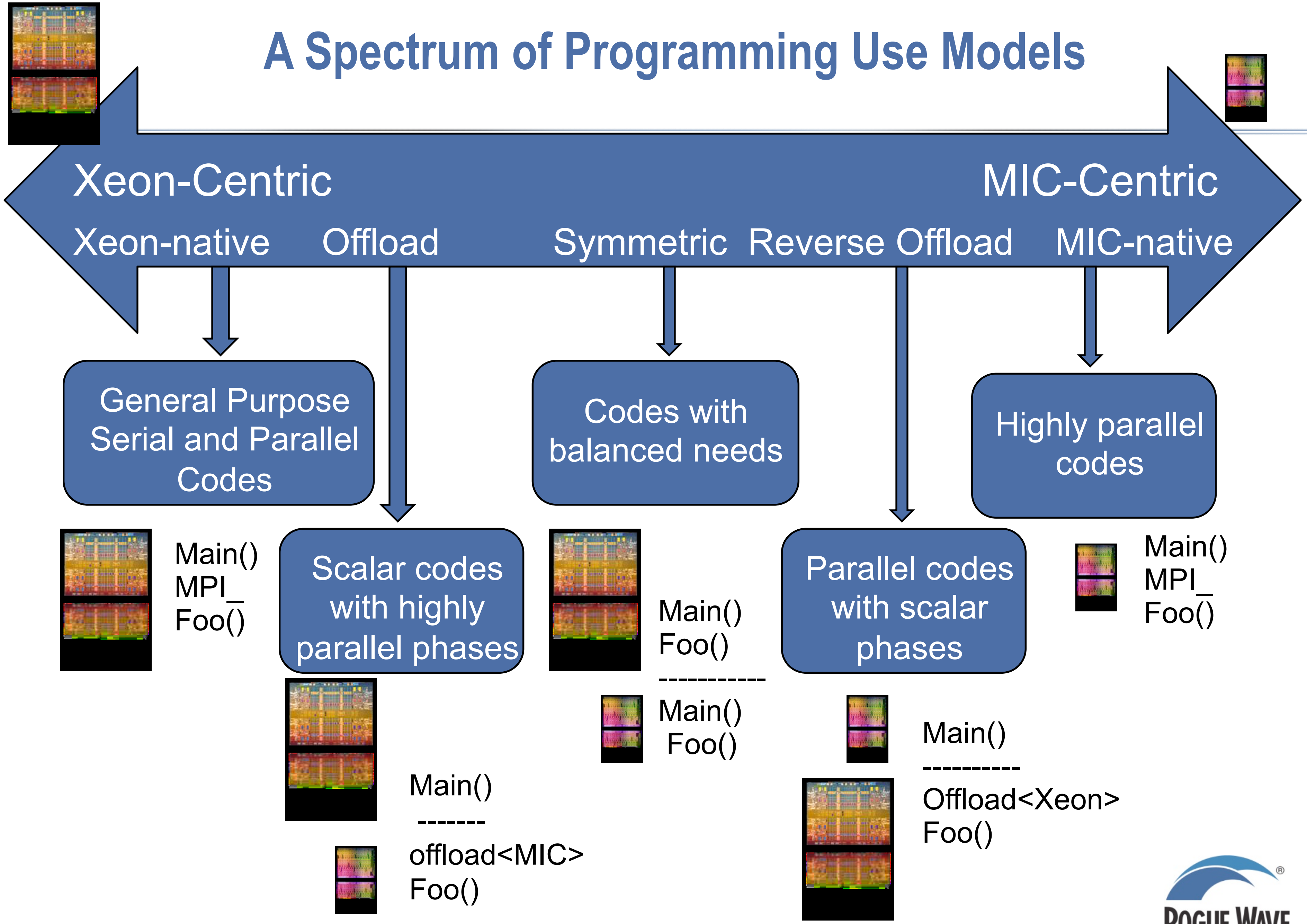
Intel Phi Debugging

with

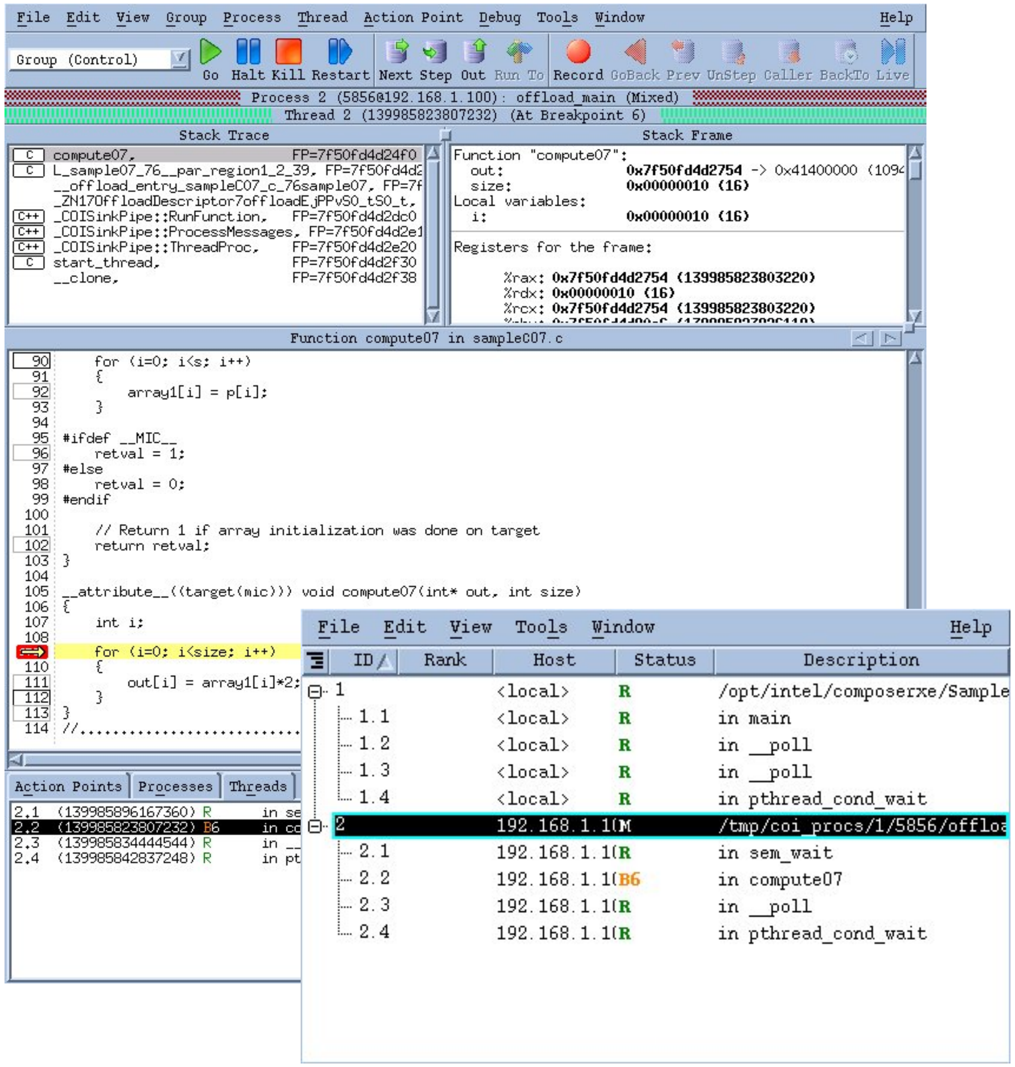
TotalView



A Spectrum of Programming Use Models


Xeon-Centric
MIC-Centric
Xeon-native
Offload
Symmetric
Reverse Offload
MIC-native
General Purpose Serial and Parallel Codes
Codes with balanced needs
Highly parallel codes
Scalar codes with highly parallel phases
Parallel codes with scalar phases
Main() MPI_ Foo()
Main() -------- offload<MIC> Foo()
Main() Foo() ------------ Main() Foo()
Main() ---------- Offload<Xeon> Foo()
Main() MPI_ Foo()

Intel MIC Port of TotalView


File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live

Process 2 (5856@192.168.1.100): offload_main (Mixed)
Thread 2 (139985823807232) (At Breakpoint 6)

Stack Trace

```
C    compute07,                             FP=7f50fd4d24f0
C    L_sample07_76__par_region1_2_39, FP=7f50fd4d2
     __offload_entry_sampleC07_c_76sample07, FP=7f
     _ZN17OffloadDescriptor7offloadEjPPvS0_tS0_t,
C++  _COISinkPipe::RunFunction,             FP=7f50fd4d2dc0
C++  _COISinkPipe::ProcessMessages, FP=7f50fd4d2e1
C++  _COISinkPipe::ThreadProc,              FP=7f50fd4d2e20
C    start_thread,                          FP=7f50fd4d2f30
     __clone,                               FP=7f50fd4d2f38
```

Stack Frame

```
Function "compute07":
  out:              0x7f50fd4d2754 -> 0x41400000 (109
  size:             0x00000010 (16)
Local variables:
  i:                0x00000010 (16)

Registers for the frame:

      %rax: 0x7f50fd4d2754 (139985823803220)
      %rdx: 0x00000010 (16)
      %rcx: 0x7f50fd4d2754 (139985823803220)
```

Function compute07 in sampleC07.c

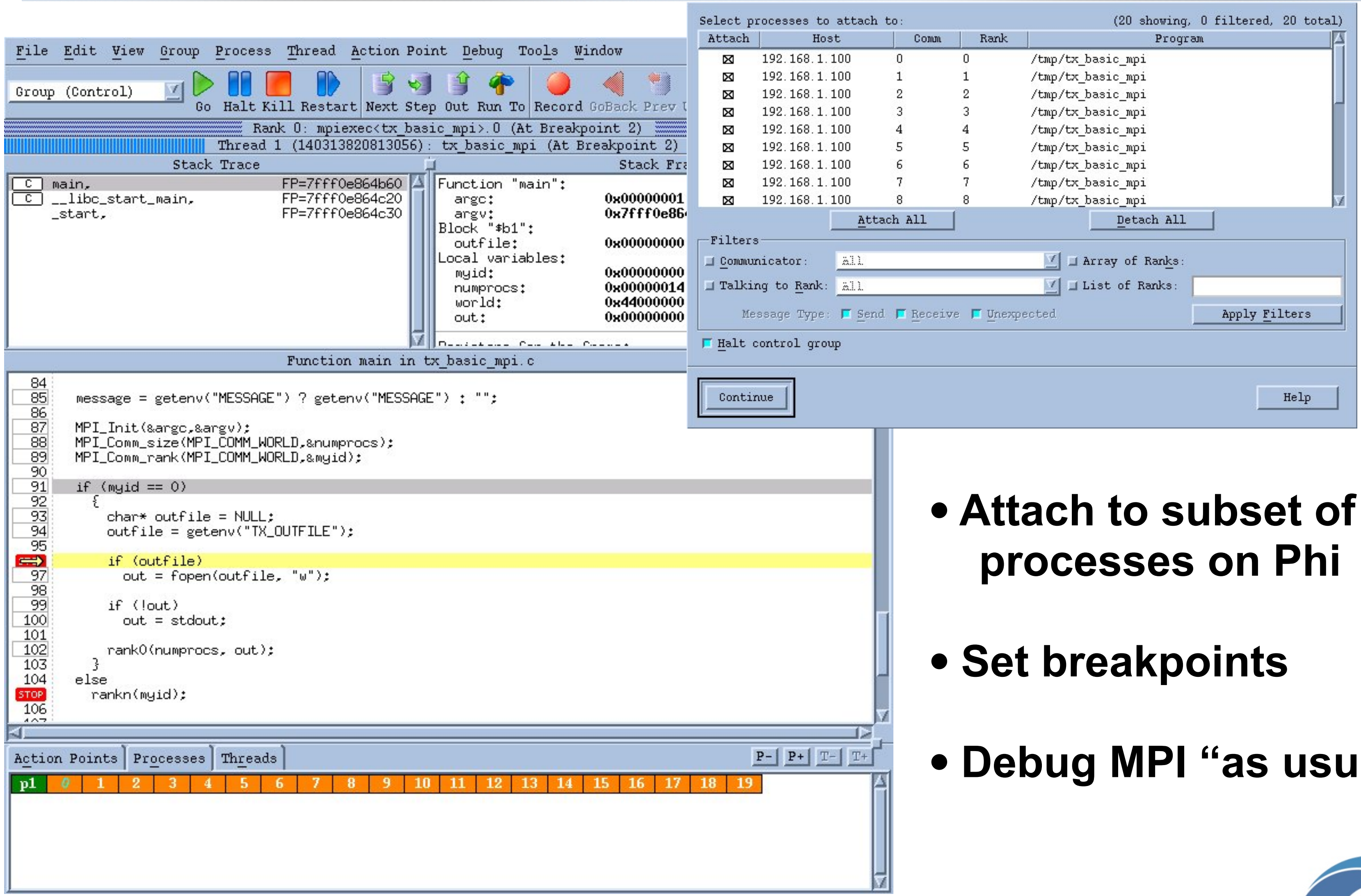
```
    for (i=0; i<s; i++)
    {
        array1[i] = p[i];
    }

#ifdef __MIC__
    retval = 1;
#else
    retval = 0;
#endif

    // Return 1 if array initialization was done on target
    return retval;
}

__attribute__((target(mic))) void compute07(int* out, int size)
{
    int i;

    for (i=0; i<size; i++)
    {
        out[i] = array1[i]*2;
    }
}
//..........................
```

```
2.1  (139985896167360) R     in se
2.2  (139985823807232) B6    in cc
2.3  (139985834444544) R     in __
2.4  (139985842837248) R     in pt
```

ID	Rank	Host	Status	Description
1		<local>	R	/opt/intel/composerxe/Sample
1.1		<local>	R	in main
1.2		<local>	R	in __poll
1.3		<local>	R	in __poll
1.4		<local>	R	in pthread_cond_wait
2		192.168.1.1(	M	/tmp/coi_procs/1/5856/offloa
2.1		192.168.1.1(	R	in sem_wait
2.2		192.168.1.1(	B6	in compute07
2.3		192.168.1.1(	R	in __poll
2.4		192.168.1.1(	R	in pthread_cond_wait

- **Full visibility of both host and Phi threads**
- **Full support of MPI programs**
- **Symmetric Debugging of heterogeneous applications with offloaded code**
- **Remote Debugging of Phi-native applications**
- **Asynchronous thread control on both Xeon and Phi**

Debugging MPI Applications


File Edit View Group Process Thread Action Point Debug Tools Window
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev
Action Points Processes Threads
P- P+ T- T+
p1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Rank 0: mpiexec<tx_basic_mpi>.0 (At Breakpoint 2)
Thread 1 (140313820813056): tx_basic_mpi (At Breakpoint 2)

Stack Trace

```
C  main,               FP=7fff0e864b60
C  __libc_start_main,  FP=7fff0e864c20
   _start,             FP=7fff0e864c30
```

Stack Fra

```
Function "main":
  argc:                 0x00000001
  argv:                 0x7fff0e86
Block "$b1":
  outfile:              0x00000000
Local variables:
  myid:                 0x00000000
  numprocs:             0x00000014
  world:                0x44000000
  out:                  0x00000000
```

Function main in tx_basic_mpi.c

```c
  message = getenv("MESSAGE") ? getenv("MESSAGE") : "";

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD,&myid);

  if (myid == 0)
    {
      char* outfile = NULL;
      outfile = getenv("TX_OUTFILE");

      if (outfile)
        out = fopen(outfile, "w");

      if (!out)
        out = stdout;

      rank0(numprocs, out);
    }
  else
    rankn(myid);
```

Select processes to attach to: (20 showing, 0 filtered, 20 total)

Attach	Host	Comm	Rank	Program
☒	192.168.1.100	0	0	/tmp/tx_basic_mpi
☒	192.168.1.100	1	1	/tmp/tx_basic_mpi
☒	192.168.1.100	2	2	/tmp/tx_basic_mpi
☒	192.168.1.100	3	3	/tmp/tx_basic_mpi
☒	192.168.1.100	4	4	/tmp/tx_basic_mpi
☒	192.168.1.100	5	5	/tmp/tx_basic_mpi
☒	192.168.1.100	6	6	/tmp/tx_basic_mpi
☒	192.168.1.100	7	7	/tmp/tx_basic_mpi
☒	192.168.1.100	8	8	/tmp/tx_basic_mpi

Attach All　Detach All

Filters
- ☐ Communicator: All
- ☐ Talking to Rank: All
- ☐ Array of Ranks:
- ☐ List of Ranks:
- Message Type: ☑ Send ☑ Receive ☑ Unexpected

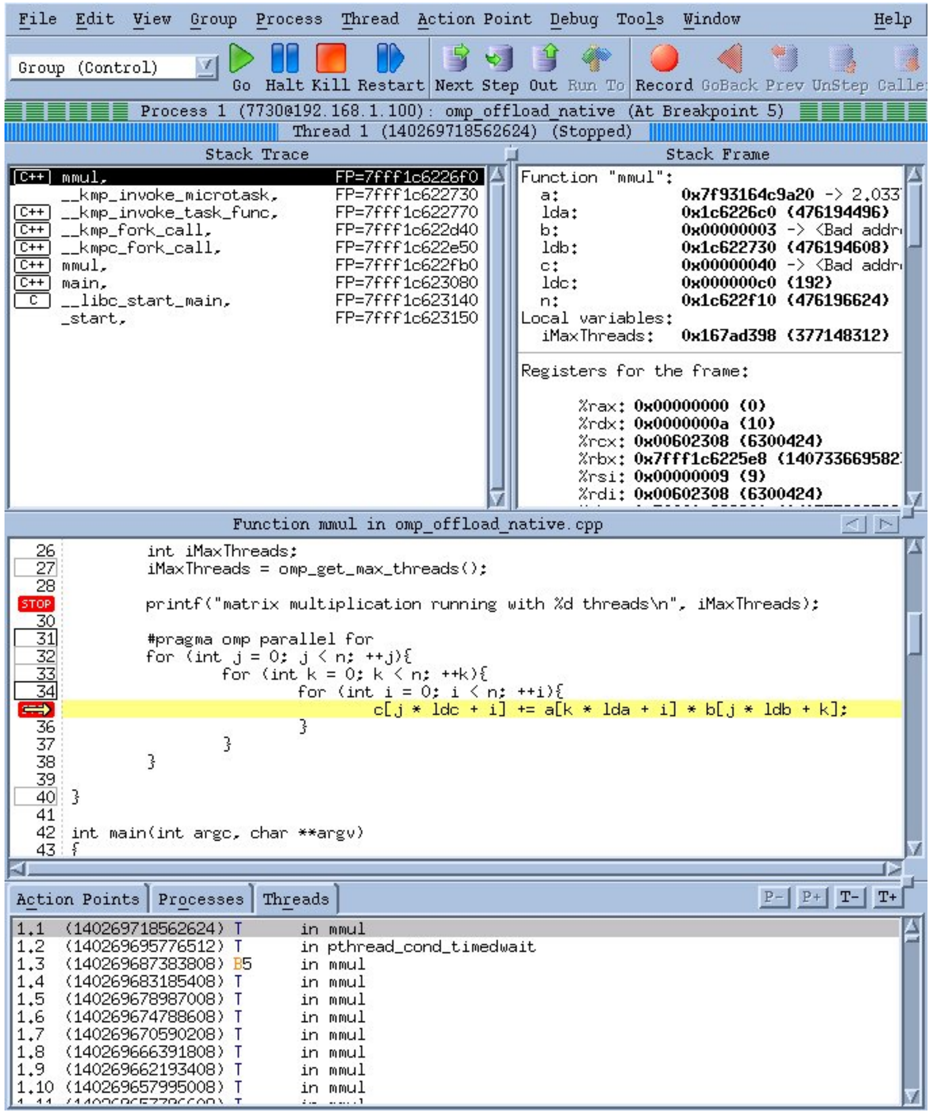
Apply Filters

☑ Halt control group

Continue　Help

- **Attach to subset of processes on Phi**
- **Set breakpoints**
- **Debug MPI "as usual"**

Remote Debugging of Applications on Phi



```
Process 1 (7730@192.168.1.100): omp_offload_native (At Breakpoint 5)
Thread 1 (140269718562624) (Stopped)
```

```
C++  mmul,                        FP=7fff1c6226f0
     __kmp_invoke_microtask,      FP=7fff1c622730
C++  __kmp_invoke_task_func,      FP=7fff1c622770
C++  __kmp_fork_call,             FP=7fff1c622d40
C++  __kmpc_fork_call,            FP=7fff1c622e50
C++  mmul,                        FP=7fff1c622fb0
C++  main,                        FP=7fff1c623080
C    __libc_start_main,           FP=7fff1c623140
     _start,                      FP=7fff1c623150
```

```
Function "mmul":
  a:            0x7f93164c9a20 -> 2.033
  lda:          0x1c6226c0 (476194496)
  b:            0x00000003 -> <Bad addr
  ldb:          0x1c622730 (476194608)
  c:            0x00000040 -> <Bad addr
  ldc:          0x000000c0 (192)
  n:            0x1c622f10 (476196624)
Local variables:
  iMaxThreads:  0x167ad398 (377148312)

Registers for the frame:

      %rax: 0x00000000 (0)
      %rdx: 0x0000000a (10)
      %rcx: 0x00602308 (6300424)
      %rbx: 0x7fff1c6225e8 (140733669582
      %rsi: 0x00000009 (9)
      %rdi: 0x00602308 (6300424)
```

Function mmul in omp_offload_native.cpp

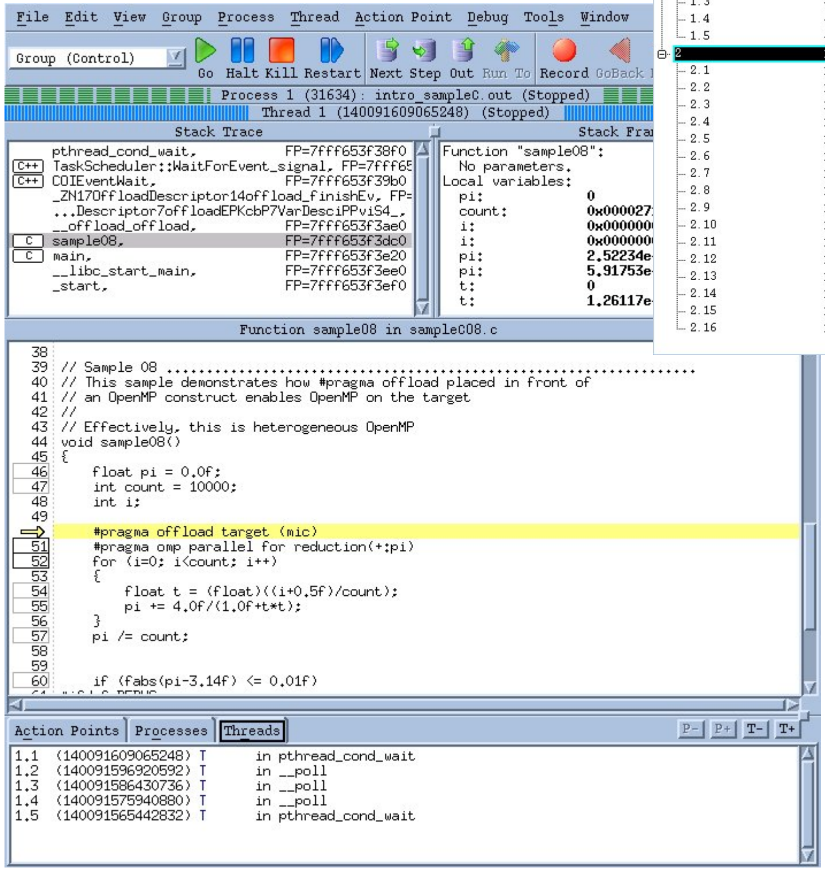
```
        int iMaxThreads;
        iMaxThreads = omp_get_max_threads();

        printf("matrix multiplication running with %d threads\n", iMaxThreads);

        #pragma omp parallel for
        for (int j = 0; j < n; ++j){
                for (int k = 0; k < n; ++k){
                        for (int i = 0; i < n; ++i){
                                c[j * ldc + i] += a[k * lda + i] * b[j * ldb + k];
                        }
                }
        }

}

int main(int argc, char **argv)
{
```

```
1.1  (140269718562624) T     in mmul
1.2  (140269695776512) T     in pthread_cond_timedwait
1.3  (140269687383808) B5    in mmul
1.4  (140269683185408) T     in mmul
1.5  (140269678987008) T     in mmul
1.6  (140269674788608) T     in mmul
1.7  (140269670590208) T     in mmul
1.8  (140269666391808) T     in mmul
1.9  (140269662193408) T     in mmul
1.10 (140269657995008) T     in mmul
```

- **Start application on Phi card**
- **Attach to running application**
- **See thread private data**
- **Investigate individual threads**
- **Kill stuck processes on Phi**

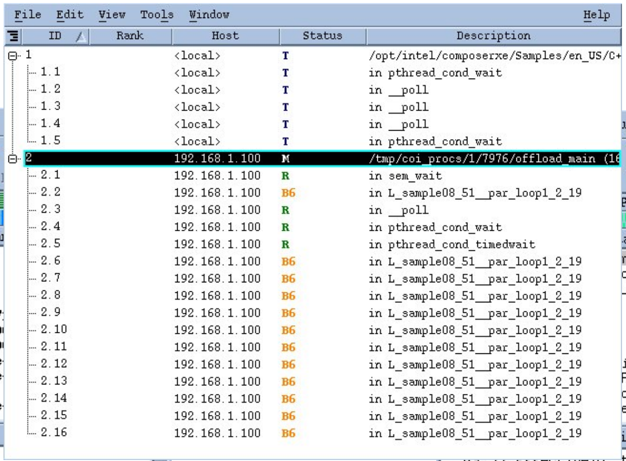
Debugging Applications with Offloaded Code

Xeon side


File Edit View Group Process Thread Action Point Debug Tools Window
Group (Control)
Go Halt Kill Restart Next Step Out Run To Record GoBack

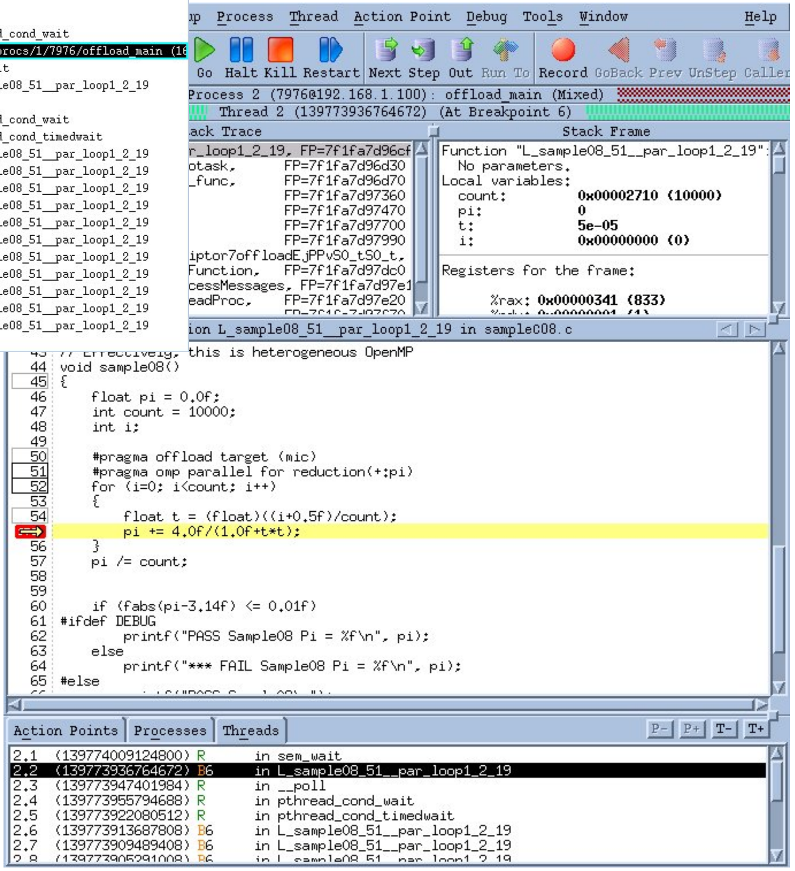
```
Process 1 (31634): intro_sampleC.out (Stopped)
Thread 1 (140091609065248) (Stopped)

Stack Trace
    pthread_cond_wait,               FP=7fff653f38f0
C++ TaskScheduler::WaitForEvent_signal, FP=7fff65
C++ COIEventWait,                    FP=7fff653f39b0
    _ZN17OffloadDescriptor14offload_finishEv, FP=
    ...Descriptor7offloadEPKcbP7VarDesciPPviS4_,
    __offload_offload,               FP=7fff653f3ae0
C   sample08,                        FP=7fff653f3dc0
C   main,                            FP=7fff653f3e20
    __libc_start_main,               FP=7fff653f3ee0
    _start,                          FP=7fff653f3ef0

Stack Frame
Function "sample08":
  No parameters.
Local variables:
  pi:        0
  count:     0x000027
  i:         0x000000
  i:         0x000000
  pi:        2.52234e
  pi:        5.91753e
  t:         0
  t:         1.26117e

Function sample08 in sample08.c
// Sample 08 ..................................................................
// This sample demonstrates how #pragma offload placed in front of
// an OpenMP construct enables OpenMP on the target
//
// Effectively, this is heterogeneous OpenMP
void sample08()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    #pragma omp parallel for reduction(+:pi)
    for (i=0; i<count; i++)
    {
        float t = (float)((i+0.5f)/count);
        pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;


    if (fabs(pi-3.14f) <= 0.01f)
```

```
1.1  (140091609065248) T    in pthread_cond_wait
1.2  (140091596920592) T    in __poll
1.3  (140091586430736) T    in __poll
1.4  (140091575940880) T    in __poll
1.5  (140091565442832) T    in pthread_cond_wait
```



ID	Rank	Host	Status	Description
1		<local>	T	/opt/intel/composerxe/Samples/en_US/C+
1.1		<local>	T	in pthread_cond_wait
1.2		<local>	T	in __poll
1.3		<local>	T	in __poll
1.4		<local>	T	in __poll
1.5		<local>	T	in pthread_cond_wait
2		192.168.1.100	M	/tmp/coi_procs/1/7976/offload_main (16
2.1		192.168.1.100	R	in sem_wait
2.2		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.3		192.168.1.100	R	in __poll
2.4		192.168.1.100	R	in pthread_cond_wait
2.5		192.168.1.100	R	in pthread_cond_timedwait
2.6		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.7		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.8		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.9		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.10		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.11		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.12		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.13		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.14		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.15		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19
2.16		192.168.1.100	B6	in L_sample08_51__par_loop1_2_19

Phi side


up Process Thread Action Point Debug Tools Window Help
Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller

```
Process 2 (7976@192.168.1.100): offload_main (Mixed)
Thread 2 (139773936764672) (At Breakpoint 6)

Stack Trace
r_loop1_2_19, FP=7f1fa7d96cf
otask,          FP=7f1fa7d96d30
_func,          FP=7f1fa7d96d70
                FP=7f1fa7d97360
                FP=7f1fa7d97470
                FP=7f1fa7d97700
                FP=7f1fa7d97990
iptor7offloadEjPPvS0_tS0_t,
Function,       FP=7f1fa7d97dc0
cessMessages, FP=7f1fa7d97e1
eadProc,        FP=7f1fa7d97e20

Stack Frame
Function "L_sample08_51__par_loop1_2_19":
  No parameters.
Local variables:
  count:      0x00002710 (10000)
  pi:         0
  t:          5e-05
  i:          0x00000000 (0)

Registers for the frame:

  %rax: 0x00000341 (833)

ion L_sample08_51__par_loop1_2_19 in sample08.c
// Effectively, this is heterogeneous OpenMP
void sample08()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    #pragma omp parallel for reduction(+:pi)
    for (i=0; i<count; i++)
    {
        float t = (float)((i+0.5f)/count);
        pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;


    if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
        printf("PASS Sample08 Pi = %f\n", pi);
    else
        printf("*** FAIL Sample08 Pi = %f\n", pi);
#else
```

```
2.1  (139774009124800) R     in sem_wait
2.2  (139773936764672) B6    in L_sample08_51__par_loop1_2_19
2.3  (139773947401984) R     in __poll
2.4  (139773955794688) R     in pthread_cond_wait
2.5  (139773922080512) R     in pthread_cond_timedwait
2.6  (139773913687808) B6    in L_sample08_51__par_loop1_2_19
2.7  (139773909489408) B6    in L_sample08_51__par_loop1_2_19
```

One debugging session for Phi-accelerated code

ROGUE WAVE SOFTWARE

TotalView provides a <u>full spectrum</u> of debugging solutions

TotalView®

Code debugging
- Highly scalable interactive GUI debugger
  - Easy to use -- without sacrificing detail that users need to debug
  - Used from workstations to the largest supercomputers
- Powerful features for debugging multi-threaded, multi-process, and MPI parallel programs
- Compatible with wide variety of compilers across several platforms and operating systems

- Memory Debugging
  - Parallel memory analysis and error detection
    - Intuitive for both intensive and infrequent users
  - Easily integrated into the validation process
- Reverse Debugging
  - Parallel record and deterministic replay within TotalView
    - Users can run their program “backwards” to find bugs
  - Allows straightforward resolution of otherwise stochastic bugs
- GPU CUDA Debugging
  - Full Hybrid Architecture Support
  - Asynchronous Warp Control
  - Multi-Device and MPI Support

ROGUE WAVE® SOFTWARE

Thanks!

ROGUE WAVE
SOFTWARE

Thank You

Download a TotalView evaluation at:

www.roguewave.com/products

ehinkel@roguewave.com